SOFTWARE DESIGN DOCUMENT
DL CSCI (4)

June, 1991

Prepared by:

BBN Systems and Technologies,
A Division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138
(617) 873-3000 FAX: (617) 873-4315

Prepared for:

Defense Advanced Research Projects Agency (DARPA)
Information and Science Technology Office
1400 Wilson Blvd., Arlington, VA 22209-2308
(202) 694-8232, AUTOVON 224-8232

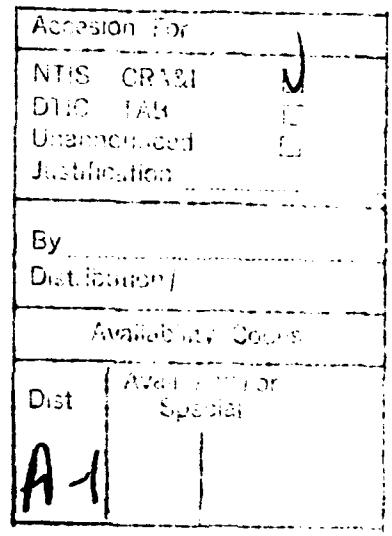
Program Manager for Training Devices (PM TRADE)
12350 Research Parkway
Orlando, FL 32826-3276
(407) 380-4518

SOFTWARE DESIGN DOCUMENT
DL CSCI (4)

June, 1991

Prepared by:

BBN Systems and Technologies,
A Division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138
(617) 873-3000 FAX: (617) 873-4315



Prepared for:

Defense Advanced Research Projects Agency (DARPA)
Information and Science Technology Office
1400 Wilson Blvd., Arlington, VA 22209-2308
(202) 694-8232, AUTOVON 224-8232

Program Manager for Training Devices (PM TRADE)
12350 Research Parkway
Orlando, FL 32826-3276
(407) 380-4518

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1991	Software Design Document

4. TITLE AND SUBTITLE

Software Design Document DL CSCI (4)

5. FUNDING NUMBERS

Contract Numbers:
MDA972-89-C-0060
MDA972-89-C-0061

6. AUTHOR(S)

Author not specified.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Bolt Beranek and Newman, Inc. (BBN)
Systems and Technologies; Advanced Simulation
10 Moulton Street
Cambridge, MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER

Advanced Simulation #:
9107

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency (DARPA
3701 North Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DARPA Report Number:
None.

11. SUPPLEMENTARY NOTES

None

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A: Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

Distribution Code:
A

13. ABSTRACT *(Maximum 200 words)*

A Simulation Network (SIMNET) project Software Design Document that describes the Data Logger (DL) Computer Software Configuration Item (CSCI number 4) of the SIMNET hardware and software training system for vehicle crew training and operational training.

14. SUBJECT TERMS

SIMNET Software Design Document for the DL CSCI (CSCI 4).

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Same as report.

Table of Contents

1 INTRODUCTION : DATA LOGGER CSCI DESCRIPTION 1
1.1 BACKGROUND 1
1.2 EXTERNAL INTERFACES 1
1.3 INTERNAL STRUCTURE 3
1.4 CONFIGURATION AND CONFIGURATION MANAGEMENT 4
1.5 TERMINOLOGY AND DOCUMENTATION 4
2 CSC DESCRIPTIONS 5
2.1 LIBRARIES 5
2.1.1 libassoc 6
2.1.2 Network Interface Access: libnetif 6
2.1.3 Logger Time Library: liblogtime 6
2.1.3.1 logtime.h 6
2.1.3.2 logtime.c 6
2.1.3.2.1 itoa 7
2.1.3.2.2 monthtoi 7
2.1.3.2.3 string_to_seconds 8
2.1.3.2.4 strings_to_datetime 8
2.1.3.2.5 datetime_to_strings 9
2.1.3.2.6 get_datetime 9
2.1.3.2.7 offset_datetime 10
2.1.4 Fast IO Library: libfio 11
2.1.4.1 Libfio Interface files 11
2.1.4.1.1 fiolib.h 11
2.1.4.1.2 fioheader.h 12
2.1.4.1.3 fioerror.h 13
2.1.4.2 Libfio files 13
2.1.4.2.1 fiofile.h 13
2.1.4.2.2 fiolib.c 13
2.1.4.2.2.1 ini_file_slots 14
2.1.4.2.2.2 find_file_slot 15
2.1.4.2.2.3 free_file_slot 15
2.1.4.2.2.4 fio_file_type 16
2.1.4.2.2.5 fio_open 17
2.1.4.2.2.6 fio_concat 18
2.1.4.2.2.7 fio_read 20
2.1.4.2.2.8 fio_write 21
2.1.4.2.2.9 fio_seek 22

2.1.4.2.2.10 fio_getrawfd 23
2.1.4.2.2.11 fio_flush 24
2.1.4.2.2.12 fio_close 24
2.1.4.2.2.13 ini_fiolib 25
2.1.4.2.3 fioheader.c 26
2.1.4.2.3.1 fio_is_null_header 26
2.1.4.2.3.2 fio_convert_header 26
2.1.4.2.3.3 fio_file_header 27
2.1.4.2.3.4 create_dphdr 28
2.1.4.2.3.5 construct_dphdr 28
2.1.4.2.3.6 fio_insert_info 29
2.1.4.2.3.7 info_field_bytes 30
2.1.4.2.3.8 find_info 31
2.1.4.2.3.9 fio_delete_info 31
2.1.4.2.3.10 fio_retrieve_info 32
2.1.4.2.3.11 set_fileid_fiohdr 33
2.1.4.2.3.12 get_fileid_fiohdr 34
2.1.4.2.3.13 set_command_fiohdr 35
2.1.4.2.3.14 get_command_fiohdr 35
2.1.4.2.3.15 get_start_date_fiohdr 36
2.1.4.2.3.16 set_tape_number_fiohdr 36
2.1.4.2.3.17 get_tape_number_fiohdr 37
2.1.4.2.3.18 set_frame_size_fiohdr 38
2.1.4.2.3.19 get_start_time_fiohdr 38
2.1.4.2.3.20 set_version_chksum_fiohdr 39
2.1.4.2.3.21 get_version_chksum_fiohdr 39
2.1.4.2.3.22 fio_print_fiohdr 40
2.1.4.2.3.23 null_fiohdr 41
2.1.4.2.4.1 fio_error 41
2.1.4.2.4.2 fio_error_string 42
2.1.4.2.5 fiocvt.h 43
2.1.4.2.6 fiocvt.c 43
2.1.4.2.6.1 cvt_fio_packet 43
2.1.4.2.6.2 fio_packet_size 44
2.1.4.2.7 fioutil.h 44
2.1.4.2.8 fioutil.c 45
2.1.4.2.8.1 fio_alloc 45
2.1.4.2.8.2 fio_free 45
2.1.4.2.9 fiodisk.h 46
2.1.4.2.10 fiodisk.c 46
2.1.4.2.10.1 next_disk_volume 47
2.1.4.2.10.2 open_UNIX_disk_file 48
2.1.4.2.10.3 fiofile_records 48

2.1.4.2.10.4 ini_disk 49
2.1.4.2.10.5 disk_open 49
2.1.4.2.10.6 disk_concat 51
2.1.4.2.10.7 disk_read 52
2.1.4.2.10.8 read_fiofile_header 53
2.1.4.2.10.9 read_dbuffer 54
2.1.4.2.10.10 read_controller 55
2.1.4.2.10.11 disk_write 56
2.1.4.2.10.12 write_fiofile_header 57
2.1.4.2.10.13 write_dbuffer 57
2.1.4.2.10.14 write_controller 58
2.1.4.2.10.15 disk_seek 59
2.1.4.2.10.16 disk_seek_time 61
2.1.4.2.10.17 volume_starttime 62
2.1.4.2.10.18 volume_endtime 62
2.1.4.2.10.19 fio_record_endtime 63
2.1.4.2.10.20 disk_getrawfd 64
2.1.4.2.10.21 disk_close 64
2.1.4.2.11 fionet.h 66
2.1.4.2.12 fionet.c 66
2.1.4.2.12.1 ini_fnet 66
2.1.4.2.12.2 fnet_open 67
2.1.4.2.12.3 fnet_read 68
2.1.4.2.12.4 fnet_write 69
2.1.4.2.12.5 fnet_flush 70
2.1.4.2.12.6 fnet_close 71
2.1.4.2.14 fiotape.h 71
2.1.4.2.14 fiotape.c 72
2.1.4.2.14.1 init_eotv_action 74
2.1.4.2.14.2 do_eotv_action 75
2.1.4.2.14.3 eotv_phase_action 75
2.1.4.2.14.4 unix_tape_action 76
2.1.4.2.14.5 promote_status_info 77
2.1.4.2.14.6 is_next_tape_volume 77
2.1.4.2.14.7 new_tbq 78
2.1.4.2.14.8 enqueue_tb 79
2.1.4.2.14.9 dequeue_tb 79
2.1.4.2.14.10 concat_tbqs 80
2.1.4.2.14.11 null_tbq 80
2.1.4.2.14.12 ini_tbuffers 80
2.1.4.2.14.13 next_available_tbuffer 81
2.1.4.2.14.14 bytes_to_trcds 82
2.1.4.2.14.15 ini_tape 83

2.1.4.2.14.16 tape_open 83
2.1.4.2.14.17 tape_concat 84
2.1.4.2.14.18 tape_read 85
2.1.4.2.14.19 read_fiofile_header 87
2.1.4.2.14.20 read_tbuffer 87
2.1.4.2.14.21 read_stream 88
2.1.4.2.14.22 read_controller 89
2.1.4.2.14.23 read_ast_handler 90
2.1.4.2.14.24 tape_write 91
2.1.4.2.14.25 write_fiofile_header 93
2.1.4.2.14.26 write_tbuffer 93
2.1.4.2.14.27 write_stream 94
2.1.4.2.14.28 write_controller 95
2.1.4.2.14.29 write_ast_handler 96
2.1.4.2.14.30 tape_seek 97
2.1.4.2.14.31 tape_seek_time 100
2.1.4.2.14.32 tvolume_available 100
2.1.4.2.14.33 tvolume_starttime 101
2.1.4.2.14.34 backup_past_target 102
2.1.4.2.14.35 tape_getrawfd 103
2.1.4.2.14.36 tape_close 104

2.2 DATA LOGGER FILES 106

2.2.1 Data Logger Header Files 106
2.2.1.1 cmc.h 106
2.2.1.2 logger.h 107
2.2.1.3 queue.h 108
2.2.1.4 racal.h 108
2.2.1.5 rtc.h 108
2.2.1.6 timer.h 109
2.2.1.7 user.h 109
2.2.1.8 util.h 109
2.2.1.9 global.h 109
2.2.1.10 logfilt.h 110

2.2.2 Data Logger Files 110
2.2.2.1 cmc.c 110
2.2.2.1.1 cmc_filter_init 111
2.2.2.1.2 cmc_protocol_types 111
2.2.2.1.3 cmc_accept 112
2.2.2.1.4 cmc_reject 112
2.2.2.1.5 cmc_init 113
2.2.2.1.6 cmc_term 114
2.2.2.1.7 cmc_zero_status 114
2.2.2.1.8 cmc_report_status 114

2.2.2.1.9 cmc_gettime16....115
2.2.2.1.10 cmc_gettime32....115
2.2.2.1.11 cmc_settime....116
2.2.2.2 logger.c....116
2.2.2.2.1 main....119
2.2.2.2.2 process_parms....119
2.2.2.2.3 init_state_variables....120
2.2.2.2.4 init_logger_variables....120
2.2.2.2.5 logger_init....121
2.2.2.2.6 init_logger_for_server....121
2.2.2.2.7 init_logger_for_copying....122
2.2.2.2.8 init_logger_for_playback....123
2.2.2.2.9 init_logger_for_recording....123
2.2.2.2.10 open_fio_files....124
2.2.2.2.11 logger_init_cmc....125
2.2.2.2.12 logger_accept_pdus....125
2.2.2.2.13 logger_get_remote_rtc....126
2.2.2.2.14 logger_put_remote_rtc....127
2.2.2.2.15 logger_fsm....127
2.2.2.2.16 logger_copying_state....128
2.2.2.2.17 init_logger_playback_state....129
2.2.2.2.18 logger_playback_state....129
2.2.2.2.19 logger_record_state....131
2.2.2.2.20 logger_reset....132
2.2.2.2.21 logger_clock_tick....132
2.2.2.2.22 logger_op_error....132
2.2.2.2.23 logger_connect....133
2.2.2.2.24 logger_continue....133
2.2.2.2.25 logger_disconnect....134
2.2.2.2.26 logger_seek_relative....134
2.2.2.2.27 logger_speed....135
2.2.2.2.28 logger_stop....136
2.2.2.2.29 logger_start....136
2.2.2.2.30 logger_suspend....137
2.2.2.2.31 logger_status....138
2.2.2.2.32 logger_get_state....138
2.2.2.2.33 logger_get_activity....139
2.2.2.2.34 ccatcher....139
2.2.2.2.35 bells....139
2.2.2.2.36 print_command_line_help....139
2.2.2.2.37 print_advice....140
2.2.2.3 queue.c....140
2.2.2.3.1 queue_of....141

2.2.2.3.2 new_queue_elt....141
2.2.2.3.3 free_queue_elt....142
2.2.2.3.4 init_queues....142
2.2.2.3.5 reinit_queues....142
2.2.2.3.6 enqueue....143
2.2.2.3.7 dequeue....143
2.2.2.3.8 queue_count....144
2.2.2.4 racal.c....144
2.2.2.4.1 racal_open....145
2.2.2.4.2 racal_close....146
2.2.2.4.3 get_from_racal....146
2.2.2.4.4 send_to_racal....146
2.2.2.4.5 racal_to_cmc....147
2.2.2.4.6 seconds_to_racal_string....147
2.2.2.4.7 get_racal_time....148
2.2.2.4.8 reset_racal_time....149
2.2.2.4.9 racal_start_recording....149
2.2.2.4.10 racal_start_playing....149
2.2.2.4.11 racal_stop....149
2.2.2.4.12 racal_seek_to_time....150
2.2.2.5 rtc.c....150
2.2.2.5.1 logger_message_location....152
2.2.2.5.2 logger_msg....152
2.2.2.5.3 logger_error....153
2.2.2.5.4 protocol_of....154
2.2.2.5.5 rtc_size....155
2.2.2.5.6 init_rtc....155
2.2.2.5.7 handle_rtc....155
2.2.2.5.8 handle_stdin_rtc....156
2.2.2.5.9 goto_handler....157
2.2.2.5.10 help_handler....157
2.2.2.5.11 info_handler....158
2.2.2.5.12 scan_handler....158
2.2.2.5.13 handle_remote_rtc....159
2.2.2.5.14 new_rtc....159
2.2.2.5.15 free_rtc....160
2.2.2.5.16 receive_rtc....160
2.2.2.5.17 send_rtc....161
2.2.2.5.18 init_apdu....161
2.2.2.5.19 init_pdu....162
2.2.2.5.20 avail_request_handler....162
2.2.2.5.21 connect_request_handler....163
2.2.2.5.22 disconnect_handler....164

2.2.2.5.23 command_request_handler 164
2.2.2.5.24 parse_start_file_info 165
2.2.2.5.25 status_request_handler 165
2.2.2.5.26 do_information 166
2.2.2.5.27 do_clock_tick 167
2.2.2.5.28 do_status_reply 167
2.2.2.5.29 send_clock_tick 168
2.2.2.6 timer.c 168
2.2.2.6.1 exe_to_cmc_time 169
2.2.2.6.2 cmc_to_exe_time 170
2.2.2.6.3 set_fast_time_factor 170
2.2.2.6.4 get_fast_time_factor 171
2.2.2.6.5 reset_time 171
2.2.2.6.6 set_time 171
2.2.2.6.7 get_time 172
2.2.2.6.8 freeze_time 172
2.2.2.6.9 unfreeze_time 173
2.2.2.6.10 on_time 173
2.2.2.6.11 time_string 174
2.2.2.7 user.c 174
2.2.2.7.1 collect_racal_time_info 176
2.2.2.7.2 collect_fiofile_info 176
2.2.2.7.3 collect_diskfile_info 177
2.2.2.7.4 collect_tapefile_info 177
2.2.2.7.5 collect_cmcsubscribe_info 178
2.2.2.7.6 collect_fioheader_info 179
2.2.2.7.7 collect_playback_info 179
2.2.2.7.8 continue_or_quit 179
2.2.2.7.9 yes_or_no 180
2.2.2.7.10 user_putstr 181
2.2.2.7.11 prompt_user 181
2.2.2.7.12 help_function 182
2.2.2.7.13 init_user 183
2.2.2.7.14 can_access_file 183
2.2.2.7.15 vlaunch 183
2.2.2.8 util.c 184
2.2.2.8.1 init_file_info 184
2.2.2.9 logfilt.c 185
2.2.2.9.1 do_init 185
2.2.2.9.2 do_packet_from_host 185
2.2.2.9.3 do_packet_from_network 185

APPENDIX A: FIO DATA FILE FORMAT A-1

APPENDIX B: PVD DATA LOGGER COMMUNICATIONS PROTOCOL B-1

INDEX BY SECTION NUMBER Index-1

1 INTRODUCTION : DATA LOGGER CSCI DESCRIPTION

1.1 Background

The Data Logger is a network management tool that captures the events of an exercise run over the network by recording the SIMNET Protocol Data Units (PDU) that were sent by all of the participants. One of the principles of distributed simulation is that individual participants in an exercise receive all the relevant data about other participants via the network. In addition, each participant is responsible for providing local state information to others via the network. Therefore, by saving the network traffic of an exercise it is possible to recreate the activity of the exercise by playing back the recorded events. This allows observers and participants to move freely through the logged exercise and re-evaluate their performance.

By analyzing the logged events, a post exercise evaluation can be performed. This data analysis activity uses the file saved by the data logger to perform off line processing on the network traffic and produce generic and tailored reports. The generic report typically attributes battlefield victories by individual participants in summary form. A tailored report could take any format, but is typically used to evaluate the detailed status and performance of an individual vehicle in the exercise.

The data logger control can be via a standard terminal interface, or remote via a client on the network. Control options include record, playback, fast-forward, rewind, and storage media (i.e. record to disk or tape).

This document is intended to provide a "roadmap" through the Data Logger code. It is divided into four sections. Section 1 will provide a high level functional description of the Data Logger CSCI. It will describe the interfaces of this CSCI with other CSCI's in the system as well as the internal structure of the Data Logger CSCI.

Section 2 will describe each of the CSC's which make up the Data Logger CSCI. Each CSC is broken down into its component CSU's based on the functionality of the code. Section 2.1 describes the libraries which contain tools and utilities that are used to interact with SIMNET. Section 2.2 contains information about the code that runs the Data Logger including the facilities that allow the person running the Data Logger to playback, record, etc.

Section 3 will describe the resources which are utilized by the Data Logger, and Section 4 will contain the source code listings.

1.2 External Interfaces

The data logger processes all external communications via the SIMNET network. The standard procedure for the logger is to be recording the data on the network, or to be playing recorded data back in a time sequenced manner. The logger can be controlled by a client, typically the Plan View Display (PVD) using a protocol specifically designed for this purpose.

When the Data Logger is used in stand alone mode, data is received from SIMNET when recording, and data is sent through SIMNET when playing back. When the Data Logger is used in server mode, data is received through SIMNET when recording, and data is sent to the client (usually the PVD) through SIMNET when playing back. Communications

between the client and the Data Logger are sent exclusively through SIMNET using a special protocol described in. Table 1.2-1. This communications scheme is shown in Figure 1.2-1.

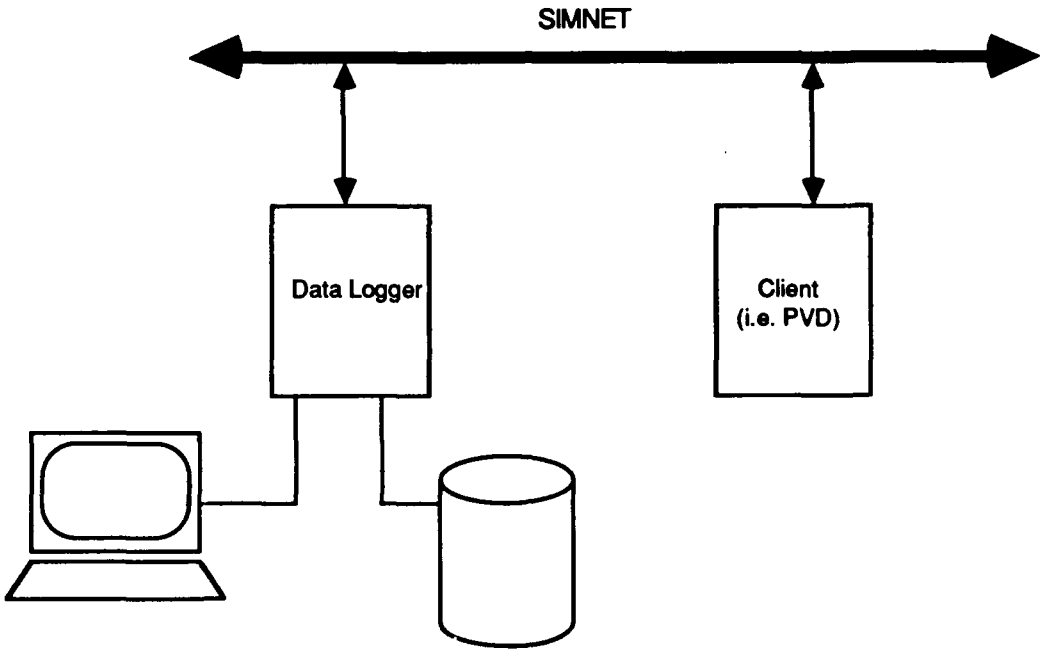


Figure 1.2-1: Communications Scheme

PDU Protocol Data Unit	CSCI Received From	CSCI Sent To	In Response To	Protocol Type	Transaction Type
loggerAvail RequestPDU Kind	Client (PVD)	Data Logger	Searching for an available Data Logger	Logger	Broadcast Datagram
logger Connect RequestPDU Kind	Client (PVD)	Data Logger	Requesting connection with specific Data Logger	Logger	ptpDatagram
logger Command Request PDUKind	Client (PVD)	Data Logger	Specifying runtime commands	Logger	ptpDatagram
loggerStatusR equest PDUKind	Client (PVD)	Data Logger	Gathering information about current operating status	Logger	ptpDatagram
logger Disconnect Request PDUKind	Client (PVD)	Data Logger	Terminate Connection of Data Logger	Logger	ptpDatagram
loggerAckPDU Kind	Data Logger	Client (PVD)	Reply to successful client requests	Logger	ptpDatagram
loggerNak PDUKind	Data Logger	Client (PVD)	Reply to Failed client requests	Logger	ptpDatagram
loggerAvail Reply PDUKind	Data Logger	Client (PVD)	Reply to client availibility requests	Logger	ptpDatagram
loggerStatus Reply PDUKind	Data Logger	Client (PVD)	Reply to client status requests	Logger	ptpDatagram
logger Information PDUKind	Data Logger	Client (PVD)	Communicate unsolicited information to the client	Logger	ptpDatagram
loggerClock TickPDUKind	Data Logger	Client (PVD)	Communicate timing information to client	Logger	ptpDatagram
loggerLast PDUKind	Data Logger	Client (PVD)		Logger	ptpDatagram

Table 1.2-1: PDU's Used by the Data Logger

1.3 Internal Structure

When the Data Logger application is run it first interprets command line switches and allocates the system resources needed to execute. The resources vary depending on whether the exercise is being logged (or played back from) a tape or a disk file. The logger then enters a command mode where it interprets client commands such as record, play, or fast forward.

A standard 'fast input /output' library was developed to enable the logger application to service the disk or the tape in a transparent manner. This allows the same entry points to be used by the logger application regardless of the media that is accessed.

In systems that have a 9-track tape drive, a second CPU is often used to handle the tape drive processing. This allows the main process to service the network in a timely manner and not be tied up pending on the relatively slow tape device.

The structure of the Data Logger is shown in Figure 1.3-1. The Data Logger software consists of a set of shared libraries and the Data Logger files.

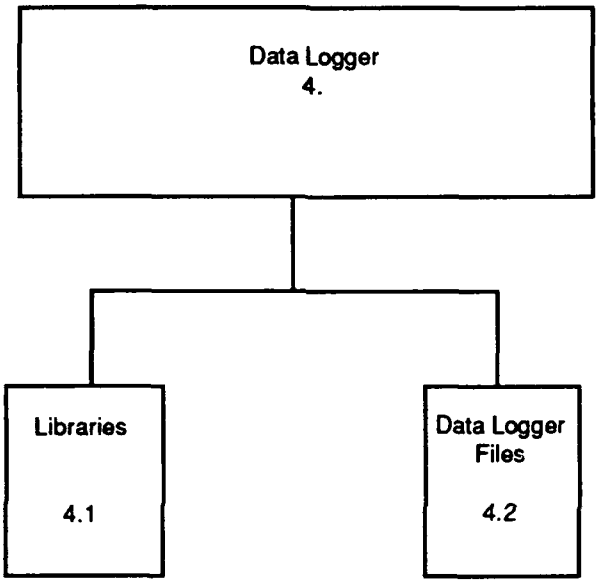


Figure 1.3-1: Data Logger CSCI Structure

The following two (2) CSC's are associated with this CSCI:

Libraries
Data Logger Files

1.4 Configuration and Configuration Management

The Data Logger program host is an MC 5600 MASSCOMP™ computer. The software for the Data Logger functions is written in C.

1.5 Terminology and Documentation

The following documents provide additional information about the Data Logger system:

- SIMNET Data Collection/Data Review, October 1989
- The SIMNET Network and Protocols, Arthur R. Pope, July, 1989

2 CSC DESCRIPTIONS

2.1 Libraries

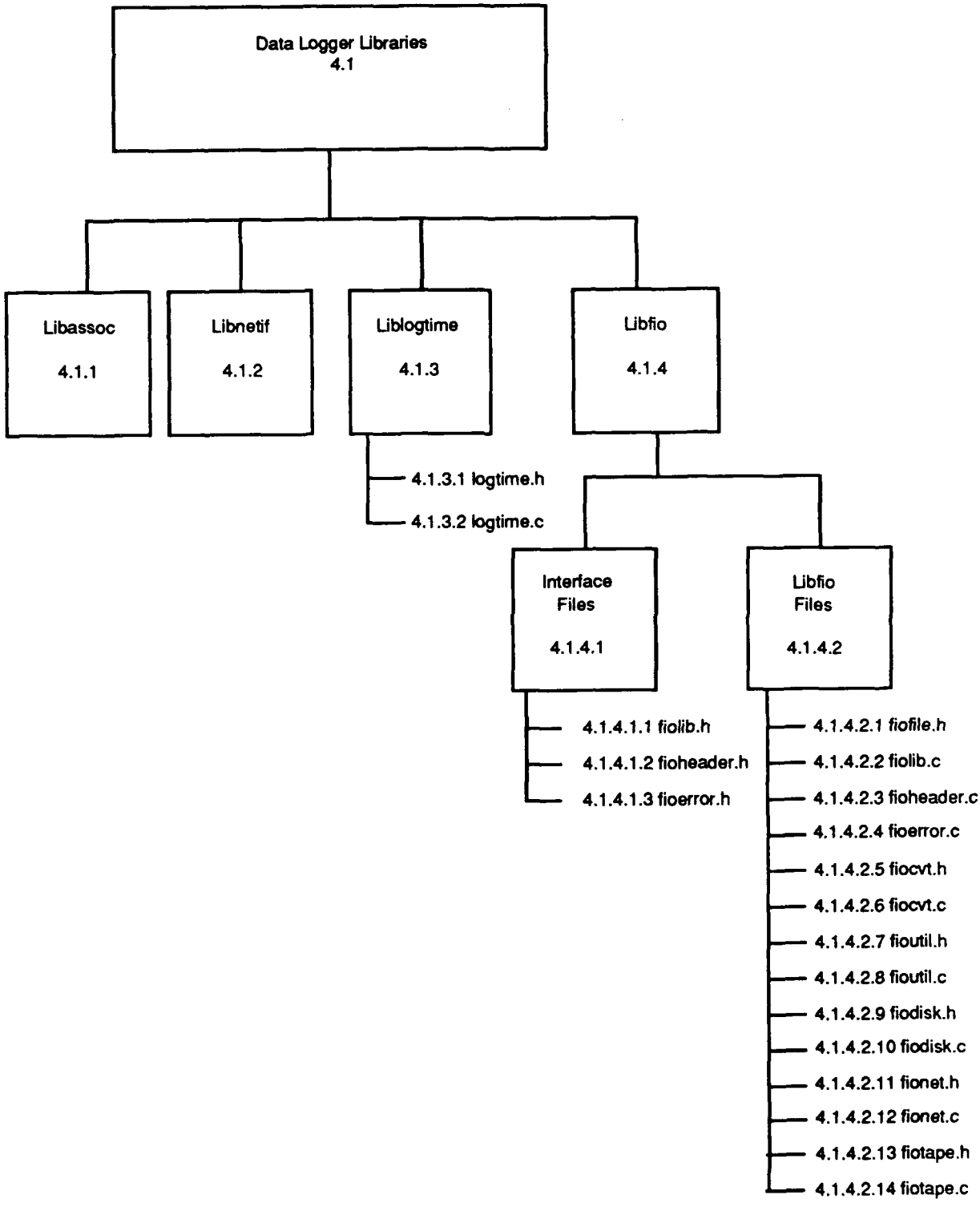


Figure 2.1-1: Data Logger Libraries Software Structure

2.1.1 libassoc

Association Layer

The SIMNET Association Layer provides network services to send/receive datagrams, and exactly once (xo) transactions. It also provides a subscription service to multicast addresses on the network, allowing an individual application to selectively accept/reject network traffic from different SIMNET exercises, and different protocols within an exercise. Libassoc communicates with the network through libnetif to provide these services. Reference Section 2.20.1 of the MCC CSCI (1) documentation.

The Data Logger does NOT use the Association Layer to handle sending/receiving datagrams. Rather it communicates directly with the layer of network services provided by the libnetif. However, it uses the Association Layer to aid in the construction of packets belonging to a special purpose protocol. The server Data Logger uses this protocol to communicate with its clients (e.g. the PVD). Reference Appendix B.

2.1.2 Network Interface Access: libnetif

Libnetif provides a high performance pathway to an unreliable datagram transport service such as that provided by an IEEE 802.3 network. Libnetif also provides several miscellaneous functions for controlling the network addapter unit. The miscellaneous functions allow applications to take advantage of adapter specific capabilities such as multicast addressing, high granularity timers, and intelligent on-board processors. Reference Section 2.20.2 of the MCC CSCI (1) documentation.

2.1.3 Logger Time Library: liblogtime

This library provides functions for converting logger timing information to and from human readable form.

2.1.3.1 logtime.h

External Procedure Declarations:

strings_to_datetime()
datetime_to_strings()
get_datetime()
offset_datetime()

2.1.3.2 logtime.c

This file defines a set of routines for handling the communication and display of logger timing information.

Includes:

<sys/types.h>
<sys/timeb.h>
<time.h>
<stdio.h>
"p_logger.h"
"logtime.h"

Imported Procedure Declarations:
timer()
strncpy()
strcpy()
strchr()

Static Procedure Declarations:
itoa()
monthtoi()

Defines:
UNKNOWN_TIMEZONE

Static Variable Declaration:
month_table[13]

2.1.3.2.1 itoa

This routine converts an integer *i* to an ASCII character *s*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
s	pointer to char	Standard
i	int	Standard

Table 2.1-1: itoa Information.

2.1.3.2.2 monthtoi

This routine converts a month *m* (of the format JAN, FEB, MAR, etc.) to an integer *i*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
m	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
i	int	The integer value of the month (1-12)
Called By		
Function	**Where Described**	
get_datetime	Sec. 2.1.3.2.6	

Table 2.1-2: monthtoi Information.

2.1.3.2.3 string_to_seconds

This routine converts a time string *time_string* (of the format: HH:MM:SS) to seconds.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
time_string	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buffer[16]	char	Standard
s	pointer to char	Standard
t	pointer to char	Standard
hours	int	Standard
minutes	int	Standard
seconds	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
3600hours + 60minutes + seconds	int	The time in seconds
-1	int	Procedure failed
Called By		
Function	**Where Described**	
collect_racal_time_info	Sec. 2.2.2.7.1	
scan_handler	Sec. 2.2.2.5.12	

Table 2.1-3: string_to_seconds Information.

2.1.3.2.4 strings_to_datetime

This routine converts a UNIX syle date/time passed in *date_string* and *time_string* (of DD-MM:YY/HH:MM:SS format) to a LoggerTime format (year, month, day, hour, minute, second, timezone, daylight savings time) in *timeptr*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
timeptr	pointer to LoggerTime	p_logger.h
date_string	pointer to char	Standard
time_string	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buffer[4]	char	Standard
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	

Table 2.1-4: strings_to_datetime Information.

2.1.3.2.5 datetime_to_strings

This routine converts a time from LoggerTime format in *timeptr* to a UNIX style format in *date_string* and *time_string*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
timeptr	pointer to LoggerTime	p_logger.h
date_string	pointer to char	Standard
time_string	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buffer[4]	char	Standard
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	
init_logger_for_copying	Sec. 2.2.2.2.7	

Table 2.1-5: datetime_to_strings Information.

2.1.3.2.6 get_datetime

This routine gets the current local date/time from Unix in *unix_time_string* and converts it to internal LoggerTime.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ctime	pointer to array of char	Standard
unix_time_string	pointer to char	Standard
clock	long	Standard
buffer	array of char	Standard
Calls		
Function	**Where Described**	
monthtoi	Sec. 2.1.3.2.2	
Called By		
Function	**Where Described**	
init_logger_for_copying	Sec. 2.2.2.2.7	
init_logger_for_recording	Sec. 2.2.2.2.9	

Table 2.1-6: get_datetime Information.

2.1.3.2.7 offset_datetime

This routine adds the specified number of *seconds* to the supplied LoggerTime (*timeptr*).

Parameters		
Parameter	**Type**	**Where Typedef Declared**
timeptr	pointer to LoggerTime	p_logger.h
seconds	unsigned	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sec	int	Standard
min	int	Standard
hr	int	Standard
dy	int	Standard
carry	int	Standard

Table 2.1-7: offset_datetime Information.

2.1.4 Fast IO Library: libfio

libfio is a high performance, packet-oriented Input/Output library for reading and writing SIMNET data files. It is based on the UNIX file Input/Output library "open", "read", "write", and "close" level. The libfio provides a unified interface for handling SIMNET simulation and voice packets on diverse mediums including the SIMNET network, disk, or 9-track tape. It provides commands for opening, reading, writing, seeking, and closing SIMNET data files. In addition, it provides a function for concatenating disk or tape files into a single logical unit. This is particularly useful with the dual tape Data Logger. Tape concatenation enables the recording of long exercises where the amount of data is expected to exceed the capacity of a single tape reel.

The fiolib provides real time I/O at data rates on the order of 225 Kbytes per second (about 1300 packets per second). The fiolib is able to support I/O either in blocking or non-blocking mode.

2.1.4.1 Libfio Interface files

The libfio interface files constitute the set of header files which must be included in applications using the libfio. One of the functions provided by these header files is the definitions of the SIMNET data file components. Reference Appendix A for a mapping of the SIMNET data file structure.

2.1.4.1.1 fiolib.h

This file defines the interface to the Fast I/O Library SIMNET data file structure. Reference Appendix A for a mapping of the SIMNET data file structure.

Procedural entry point declarations:

fio_file_type()
fio_open()
fio_concat()
fio_read()
fio_write()
fio_seek()
fio_getrawfd()
fio_flush()
fio_close()

Flags, parameter values, and macros for the procedural entry points are defined:

Defines for fio_file_type:

FIO_CMC_NETWORK	-network file
FIO_NINETRACK_TAPE	-nine-track tape file
FIO_CARTRIDGE_TAPE	-Not Supported
FIO_FLOPPY_TAPE	-Not Supported
FIO_DISK	-disk file
FIO_NULL_TYPE	-NULL file

Flags and Macros for fio_open:

FIO_APPEND	
FIO_CTG	-disk files must be contiguous for effeciency during recording (disk files only)
FIO_LOOP	-for debugging purposes
FIO_NBLOCK	-non-blocking mode
FIO_PROMISCUOUS	-pay attention to everything on the network
FIO_RDONLY	-read only
FIO_RDWR	-read, write mode (net files only)
FIO_REWOFFL	-rewind off-line
FIO_VERSION1	-for conversion purposes; this is a version 1 file
FIO_VERSION2	-for conversion purposes; this is a version 2 file
FIO_WRONLY	-write only
IS_CONTINUOUS()	
IS_LOOPING()	
IS_NBLOCKING()	
IS_READABLE()	
IS_VERSION1()	
IS_VERSION2()	
IS_VERSION()	
IS_WRITABLE	

For fio_open *size* parameter:

FIOSIZE_600	-600' 9-track tape
FIOSIZE_1200	-1200' 9-track tape
FIOSIZE_2400	-2400' 9-track tape
FIOSIZE_TINY	-for debugging purposes
FIOSIZE_SMALL	-for debugging purposes

For fio_flush *flags* parameter:

FIOFLUSH_RECEIVE
FIOFLUSH_TRANSMIT

For fio_seek *whence* parameter:

FIOSEEK_SET
FIOSEEK_CUR
FIOSEEK_END

For fio_seek and fio_getrawfd *vol* parameter:

FIO_CURRENT_FD

2.1.4.1.2 fioheader.h

This is the interface file for routines accessing fio_header type abstractions. Every SIMNET data file consists of a fio_header followed by an arbitrary number of fixed length file records. The fio_header contains information to be used by DataProbe while collecting statistical information. In addition, it contains a count of the number of file records in the file, various version information fields, and a table of fio_packet time stamps for seeking in the file by time. Reference Appendix A for a mapping of the SIMNET data file and header. This file defines the structure of the header record for SIMNET data files as well as declares the external procedures for manipulating the header.

2.1.4.1.3 fioerror.h

This file provides an error handling capability for applications using the fiolib. It defines a set of error conditions and declares the set of procedures to access the associated error messages. This file declares the procedures defined in "fioerror.c", **fio_error()**, **fio_error_string()**, and lists the fiolib error definition codes.

2.1.4.2 Libfio files

These files make up the libfio.

2.1.4.2.1 fiofile.h

This file contains definitions of the data types of the SIMNET data file and header structures and declares certain variables as these types.

Version 1 files contain no fio header and have fixed length packets.
Version 2 files contain an fioheader, use variable length packets, and use Ethernet version 2 protocol.
Version 3-5 files contain no changes
Version 6 files use Ethernet 802.3 Ethernet protocol.

FIO_SLOT_E	fio_slot_e
PHYS_TYPE_E	phys_e
FIO_FILE_INFO	file_info
FIO_FILE	fio_file_t, fio_file_p
FIO_RECORD_HEADER	fio_record_header
FIO_RECORD	*fio_record*

Defines of constants and macros:

FIOLIB_VERSION2_CHKSUM	-- Version 2 file.
FIOLIB_VERSION6_CHKSUM	-- Version 6 file.
FIOLIB_VERSION_CHKSUM	
FIOLIB_MAJOR_REVNUM	-- Major revision number of libfio.
FIOLIB_MINOR_REVNUM	-- Minor revision number of libfio
FIO_RECORD_BYTES	
FIO_RECORD_BYTES_VERSION1_DISK	
FIO_RECORD_BYTES_VERSION1_TAPE	
FIO_RECORD_PACKET_BYTES	
BLOCK_SIZE	
FIO_RECORD_BLOCKS	
FIO_RECORD_PACKETS ()	
FIO_RECORD_BYTE_COUNT ()	

2.1.4.2.2 fiolib.c

This file along with "fioerror.c" and "fioheader.c" contain the set of external procedures which constitute the interface to the libfio. They dispatch specific routines to media depending on the type of the specified file handle.

Includes:
<stdio.h>
<errno.h>
<sys/types.h>
"fiolib.h"
"fioerror.h"
"fiofile.h"
"fioheader.h"
"fiotape.h"
"fionet.h"
"fiodisk.h"

Declarations:
fio_volume_number
fio_seek_volume
fio_seek_time
fio_file_t
device_types
fio_lib_initialized

Static procedure declarations:
ini_fiolib()

External procedure declarations:
fopen()

Constant and Macro defines:
FIO_MAX_OPEN
FIOPTR_OF()
IS_VALID_FD()
CONTIGUOUS()
FIOTYPE()
FIO_DEVICE_NAMES_FILE

2.1.4.2.2.1 ini_file_slots

ini_file_slots initializes the open_files array. Each of the 25 files (the value of FIO_MAX_OPEN, which is defined externally) is initialized with the following characteristics:
state: FIO_SLOT_AVAIL
type: FIO_NULL_TYPE
open_flags: 0
curvol: -1
info.iptr: NULL

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
fd	fio_desc	Sec. 2.1.4.1.1 fiolib.h

Called By	
Function	**Where Described**
ini_fiolib	Sec. 2.1.4.2.2.13

Table 2.1-8: ini_file_slots Information.

2.1.4.2.2.2 find_file_slot

find_file_slot finds an available slot in the open files array and returns the index of the first available open file descriptor. This procedure checks for the the lowest indexed file with a state of FIO_SLOT_AVAIL, changes the state to FIO_SLOT_UNAVAIL, and returns the file index in *fd*.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
fd	fio_desc	Sec. 2.1.4.1.1 fiolib.h
Return Values		
Return Value	**Type**	**Meaning**
fd	fio_desc	The index of the first available member of the open_file array.
Called By		
Function	**Where Described**	
fio_open	Sec. 2.1.4.2.2.5	

Table 2.1-9: find_file_slot Information.

2.1.4.2.2.3 free_file_slot

The first available slot in the open_files array was determined in **find_file_slot**() and returned in *fd*. **free_file_slot**() frees the file slot passed in *fd* by giving it the following file characteristics:

state: FIO_SLOT_AVAIL
type: FIO_NULL_TYPE
open_flag: 0
curvol: -1
info.iptr: NULL

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_desc	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fdptr	register fio_file_p	Sec. 2.1.4.2.1 fiofile.h

Calls	
Function	**Where Described**
FIOPTR_OF	Sec. 2.1.4.2.2 fiolib.c (macro definition)

Table 2.1-10: free_file_slot Information.

2.1.4.2.2.4 fio_file_type

This procedure checks the path name of the file to determine the file type. The device name choices are:

"cmc" -network
"9tp" -nine track tape
"ctp" -cartridge tape (not supported)
"ftp" -floppy tape (not supported)
"dsk" -disk
NULL

This procedure defines BUFFER_SIZE as 32.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
path	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buffer [BUFFER_SIZE]	char array	Standard
devname [BUFFER_SIZE]	char array	Standard
devtype [BUFFER_SIZE]	char array	Standard
fd	pointer to FILE	Standard
i	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_DISK	int	The integer code for the disk device type
i	int	The integer code for a device other than disk
FIO_ERROR	int	The procedure failed
Errors		
Error		**Reason for Error**
FIO_DEVICE_FILE_ERROR		The devices file '/simnet/data/fiodevs' is missing
FIO_DEVICE_TYPE_ERROR		Error in '/simnet/data/fiodevs' for the specified file
FIO_DEVICE_NAME_ERROR		The specified device name is an unsupported UNIX device
Called By		
Function		**Where Described**
fio_open		Sec. 2.1.4.2.2.5

Table 2.1-11: fio_file_type Information.

2.1.4.2.2.5 fio_open

fio_open returns the file descriptor, *fd*, for a named file which can then be used by the calling process for input/output operations. The *path* argument points to a null terminated string containing the pathname of the file to be opened. The path can refer to a tape drive, a disk file, or a network name. The *oflag* argument specifies the file status flags associated with the file descriptor and can be constructed by *or*ing together flags from the following list. Exactly one from the first two flags must be specified:

FIO_RDONLY — Open for reading only.

FIO_WRONLY — Open for writing only.

FIO_CREATE — If the file exists, this flag has no effect. Otherwise, this flag causes **fio_open**() to create a new file.

FIO_CTG — Verify or create a contiguous disk file. This flag requires that the optional *size* argument be supplied. The specified file is verified to be contiguous and at least the specified number of fio_packets. If a new file is being created (the FIO_CREATE flag is set), *size* fio_packets of continuous space are allocated to it. Any call to **fio_write**() which would result in the file growing larger than *size* fio_packets will fail and return an error. This flag need not be set to open an existing contiguous file unless the file size is to be verified.

FIO_BLOCK — Specify synchronous I/O.

FIO_APPEND — **fio_write**() will append the file.

For disk files, the *size* argument is required only when the FIO_CTG flag is used. In this instance, the *size* parameter refers to the minimum size of the file in bytes. For 9-track tape files, the *size* argument is required using one of the following defines:

FIO_SIZE_600 -- 600' 9-track tape
FIO_SIZE_1200 -- 1200' 9-track tape
FIO_SIZE_2400 -- 2400' 9-track tape

Upon successful completion, a non-negative file descriptor *fd* is returned. In the event of an error, FIO_ERROR is returned.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
path	pointer to char	Standard
oflag	int	Standard
size	unsigned long	Standard
header	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
type	int	Standard
status	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	fio_descr	An error is returned
fd	fio_descr	*fd* is the new file descriptor for a named file which can be used by the calling process for input/output operations

Errors	
Error	**Reason for Error**
FIO_MAX_OPEN_ERROR	The maximum number of open files has been exceeded.
FIO_INVALID_RDWR_FLAGS_ERROR	The file has invalid FIO_RDONLY and FIO_WRONLY flags
FIO_INVALID_VERSION_FLAG_ERROR	Data file has invalid version id; cannot be opened for writing
FIO_DEVICE_FILE_TYPE_ERROR	The specified UNIX device file type is not supported in this version
FIO_DEVICE_FILE_ERROR	The devices file 'simnet/data/fiodevs' is missing

Calls	
Function	**Where Described**
ini_fiolib	Sec. 2.1.4.2.2.13
find_file_slot	Sec. 2.1.4.2.2.2
FIOPTR_OF	Sec. 2.1.4.2.2 fiolib.c (macro definition)
fio_file_type	Sec. 2.1.4.2.2.4
IS_READABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)

Called By	
Function	**Where Described**
open_fio_files	Sec. 2.2.2.2.10
logger_init_cmc	Sec. 2.2.2.2.11

Table 2.1-12: fio_open Information.

2.1.4.2.2.6 fio_concat

This procedure concatenates the two disk files or two tape files into a single logical file entity. By using two tape drives and switching tapes as soon as the end of each physical tape is reached, these commands can be used to handle fio_packet sequences of arbitrary length. If this is done, however, it will cause some special problems for **fio_seek**() since it may not be possible to seek to the logical beginning or end of the concatenated file without operator intervention. Therefore, in the case of concatenated tape files

FIOSEEK_SET and FIOSEEK_END are relative to the mounted portion of the concatenated file.

fio_concat() concatenates the file represented by *path* to the open file descriptor, *fd*. **fio_concat**() accepts only tape files or disk files, and both component files must be of the same type. The file represented by *path* inherits all its characteristics except *size* from the open file. **fio_concat**() returns a new file descriptor upon successful completion. If for some reason the concat fails, the procedure returns FIO_ERROR.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h
path	pointer to a char	Standard
size	unsigned long	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
type	int	Standard
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	fio_descr	The procedure failed
fd	fio_descr	the new file descriptor for the concatenated file
Errors		
Error		**Reason for Error**
FIO_INVALID_FD_ERROR		The file descriptor does not correspond to an OPEN file
FIO_CONFLICTING_FILE_TYPES_ERROR		The file descriptors are not of the same type
FIO_DEVICE_FILE_TYPE_ERROR		The specified UNIX device file type is not supported in this version
Calls		
Function		**Where Described**
IS_VALID_FD		Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOPTR_OF		Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOTYPE		Sec. 2.1.4.2.2 fiolib.c (macro definition)
tape_concat		Sec. 2.1.4.2.14.17
disk_concat		Sec. 2.1.4.2.10.6
Called By		
Function		**Where Described**
open_fio_files		Sec. 2.2.2.2.10

Table 2.1-13: fio_concat Information.

2.1.4.2.2.7 fio_read

fio_read() attempts to read a data packet *pkt* from the object referenced by the descriptor *fd* to an internal buffer. *pkt* is a pointer to a pointer to an FIO_PACKET. If *copy* is TRUE then **fio_read** assumes *pkt* is a pointer to a buffer which is large enough for the largest simulation packet, and copies the next packet in the specified SIMNET data file. If *copy* is FALSE, **fio_read** merely assigns the address of the next simulation packet to *pkt*. This pointer is guaranteed to be valid until the next time **fio_read** is called. This routine returns FIO_OK upon successful completion and FIO_ERROR in the event of an error.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h
pkt	pointer to a pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
copy	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	fio_descr	The procedure failed
FIO_OK	fio_descr	The procedure was successful
Errors		
Error		**Reason for Error**
FIO_INVALID_FD_ERROR		The file descriptor does not correspond to an OPEN file
FIO_NOT_READABLE_ERROR		The file is not open for reading
FIO_DEVICE_FILE_TYPE_ERROR		The specified UNIX device file type is not supported in this version
Calls		
Function		**Where Described**
IS_VALID_FD		Sec. 2.1.4. 2.1 fiolib.c (macro definition)
FIOPTR_OF		Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOTYPE		Sec. 2.1.4.2.2 fiolib.c (macro definition)
IS_READABLE		Sec. 2.1.4.1.1 fiolib.h (macro definition)
fnet_read		Sec. 2.1.4.2.12.3
tape_read		Sec. 2.1.4.2.14.18
disk_read		Sec. 2.1.4.2.10.7
Called By		
Function		**Where Described**
logger_get_remote_rtc		Sec. 2.2.2.2.13
logger_copying_state		Sec. 2.2.2.2.16
logger_playback_state		Sec. 2.2.2.2.18
logger_record_state		Sec. 2.2.2.2.19

Table 2.1-14: fio_read Information.

2.1.4.2.2.8 fio_write

fio_write() attempts to write a data fio_packet *pkt* to the object referenced by the descriptor *fd*.

If the file was opened with the FIO_APPEND flag set, the file pointer will be set to the end of the file prior to each write. This routine returns FIO_OK upon successful completion, and FIO_ERROR in the event of an error.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h

Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	fio_descr	The procedure failed
FIO_OK	fio_descr	The procedure was successful

Errors	
Error	**Reason for Error**
FIO_INVALID_FD_ERROR	The file descriptor does not correspond to an OPEN file
FIO_NOT_WRITABLE_ERROR	The file is not open for writing
FIO_DEVICE_FILE_TYPE_ERROR	The specified UNIX device file type is not supported in this version

Calls	
Function	**Where Described**
IS_VALID_FD	Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOPTR_OF	Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOTYPE	Sec. 2.1.4.2.2 fiolib.c (macro definition)
IS_WRITABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
fnet_write	Sec. 2.1.4.2.12.4
tape_write	Sec. 2.1.4.2.14.24
disk_write	Sec. 2.1.4.2.10.11

Called By	
Function	**Where Described**
logger_put_remote_rtc	Sec. 2.2.2.2.14
logger_copying_state	Sec. 2.2.2.2.16
logger_playback_state	Sec. 2.2.2.2.18
logger_record_state	Sec. 2.2.2.2.19

Table 2.1-15: fio_write Information.

2.1.4.2.2.9 fio_seek

This routine seeks to a specified location. *fd* is a file descriptor returned from an **fio_open**() for a disk file or a tape file. **fio_seek** is valid for disk and tape files only. **fio_seek**() sets the file pointer associated with *fd* to the packet specified by the *offset* and *whence* arguments. Symbolic constants for *whence* are defined as follows:

FIOSEEK_SET Set the file pointer to *offset* seconds in the exercise

FIOSEEK_CUR Set the file pointer to current location plus *offset* seconds

FIOSEEK_END Set the file pointer to End-Of-File minus*offset* seconds

fio_seek() returns FIO_OK upon successful completion, and FIO_ERROR in the event of an error.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h
offset	int	Standard
whence	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h

Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	fio_desc	The procedure failed
FIO_OK	fio_desc	The procedure was successful

Errors	
Error	**Reason for Error**
FIO_INVALID_FD_ERROR	The file descriptor does not correspond to an OPEN file
FIO_SEEK_WRITABLE_FILE_ERROR	Seeking is not allowed in files opened for writing
FIO_DEVICE_FILE_TYPE_ERROR	The specified UNIX device file type is not supported in this version

Calls	
Function	**Where Described**
IS_VALID_FD	Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOPTR_OF	Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOTYPE	Sec. 2.1.4.2.2 fiolib.c (macro definition)
IS_WRITABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
tape_seek	Sec. 2.1.4.2.14.30
tape_seek_time	Sec. 2.1.4.2.14.31
disk_seek	Sec. 2.1.4.2.10.15
disk_seek_time	Sec. 2.1.4.2.10.16

Called By	
Function	**Where Described**
logger_seek_relative	Sec. 2.2.2.2.26

Table 2.1-16: fio_seek Information.

2.1.4.2.2.10 fio_getrawfd

This routine gets the UNIX file descriptor associated with the fiolib file descriptor *fd*. The returned file descriptor can then be used in Simnet Network Library calls such as **net_filter()**. In the event of an error, **fio_getrawfd()** returns FIO_ERROR. Extreme care should be taken when using these calls since they are taking place outside the scope and therefore outside the control of the fiolib. Using UNIX system calls is particularly risky. This command is intended to supply the user with an escape to the more esoteric Simnet network commands which are not supported by the fiolib. If you find it necessary to call UNIX commands, then either the fiolib interface should be expanded, or you should not be using fiolib.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h
vol_offset	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	fio_descr	The procedure failed
retval	int	The raw tape or disk file descriptor
Errors		
Error		**Reason for Error**
FIO_INVALID_FD_ERROR		The file descriptor does not correspond to an OPEN file
FIO_DEVICE_FILE_TYPE_ERROR		The specified UNIX device file type is not supported in this version
Calls		
Function		**Where Described**
IS_VALID_FD		Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOPTR_OF		Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOTYPE		Sec. 2.1.4.2.2 fiolib.c (macro definition)
tape_getrawfd		Sec. 2.1.4.2.14.35
disk_getrawfd		Sec. 2.1.4.2.10.20

Table 2.1-17: fio_getrawfd Information.

2.1.4.2.2.11 fio_flush

fio_flush() is valid for network files only. **net_flush** in the libnetif is called to flush the transmit or receive buffers. The *flags* argument is used to specify which of these buffers is to be affected: FIOFLUSH_TRANSMIT or FIOFLUSH_RECEIVE. *flags* is a bit vector.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h
flags	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
status	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	fio_desc	The procedure failed
FIO_OK	int	The procedure was successful

Errors	
Error	**Reason for Error**
FIO_INVALID_FD_ERROR	The file descriptor does not correspond to an OPEN file
FIO_INTERNAL_ERROR	fio_flush is not supported for tape files
FIO_DEVICE_FILE_TYPE_ERROR	The specified UNIX device file type is not supported in this version

Calls	
Function	**Where Described**
IS_VALID_FD	Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOPTR_OF	Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOTYPE	Sec. 2.1.4.2.2 fiolib.c (macro definition)
fnet_flush	Sec. 2.1.4.2.12.5
disk_getrawfed	Sec. 2.1.4.2.10.20

Called By	
Function	**Where Described**
logger_continue	Sec. 2.2.2.2.24
logger_seek_relative	Sec. 2.2.2.2.26
logger_start	Sec. 2.2.2.2.29

Table 2.1-18: fio_flush Information.

2.1.4.2.2.12 fio_close

This routine closes the file specified by *fd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	fio_descr	Sec. 2.1.4.1.1 fiolib.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
status	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_ERROR	int	An error is returned
FIO_OK	int	The procedure was successful
Errors		
Error		**Reason for Error**
FIO_INVALID_FD_ERROR		The file descriptor does not correspond to an OPEN file
FIO_DEVICE_FILE_TYPE_ERROR		The specified UNIX device file type is not supported in this version
Calls		
Function		**Where Described**
IS_VALID_FD		Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOPTR_OF		Sec. 2.1.4.2.2 fiolib.c (macro definition)
FIOTYPE		Sec. 2.1.4.2.2 fiolib.c (macro definition)
fnet_close		Sec. 2.1.4.2.12.6
tape_close		Sec. 2.1.4.2.14.36
disk_close		Sec. 2.1.4.2.10.21
Called By		
Function		**Where Described**
open_fio_files		Sec. 2.2.2.2.10
logger_reset		Sec. 2.2.2.2.20

Table 2.1-19: fio_close Information.

2.1.4.2.2.13 ini_fiolib

This routine initializes an available slot in the open files array.

Calls	
Function	**Where Described**
ini_file_slots	Sec. 2.1.4.2.2.1
ini_disk	Sec. 2.1.4.2.10.4
ini_fnet	Sec. 2.1.4.2.12.1
ini_tape	Sec. 2.1.4.2.14.15
Called By	
Function	**Where Described**
fio_open	Sec. 2.1.4.2.2.5

Table 2.1-20: ini_fiolib Information.

2.1.4.2.3 fioheader.c

Every SIMNET data file starts with a 24K byte header block. The file provides routines for accessing the information in that header. This file contains external procedures for manipulating SIMNET data file headers. Reference Appedix A for a more detailed mapping of the SIMNET data header structure.

Includes:

<stdio.h>
"fiofile.h"
"fioerror.h"
"fioheader.h"
"p_logger.h"

Standard procedure declarations:

strncpy()

Static procedure declarations:

construct_dphdr()
set_frame_size_fiohdr()
find_info()

2.1.4.2.3.1 fio_is_null_header

This procedure returns a TRUE if the header specified by *file_hdr* is NULL.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
Return Values		
Return Value	**Type**	**Meaning**
TRUE	int	The header is NULL
FALSE	int	The header is not NULL
Called By		
Function	**Where Described**	
fio_is_null_header	Sec. 2.1.4.2.3.1	

Table 2.1-21: fio_is_null_header Information.

2.1.4.2.3.2 fio_convert_header

This procedure is used only when converting Version 1 style SIMNET data files to the latest format. It converts old-style headers to the style for Version 2 and later data files.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
frame_size	int	Standard
file_id	pointer to char	Standard
command_str	pointer to char	Standard

Calls	
Function	**Where Described**
set_frame_size_fiohdr	Sec. 2.1.4.2.3.18
fio_insert_info	Sec. 2.1.4.2.3.6
Called By	
Function	**Where Described**
init_logger_for_copying	Sec. 2.2.2.2.7

Table 2.1-22: fio_convert_header Information.

2.1.4.2.3.3 fio_file_header

This routine constructs an fio_header in the *file_hdr* buffer provided. *file_id*, *num_exercises*, *exercises*, *command_str*, *tape_type*, *run_number*, *project_id*, *comment*, *date_of_run*, *tape_number*, *bits_per_word*, *frame_size*, *header_size*, *date_of_copy*, *time_of_run*, and *time_of_launch* are the fio_header fields passed in.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
file_id	pointer to char	Standard
num_exercises	short	Standard
exercises	pointer to char	Standard
command_str	pointer to char	Standard
tape_type	pointer to char	Standard
run_number	int	Standard
project_id	pointer to char	Standard
comment	pointer to char	Standard
date_of_run	pointer to char	Standard
tape_number	int	Standard
bits_per_word	int	Standard
frame_size	int	Standard
header_size	int	Standard
date_of_copy	pointer to char	Standard
time_of_run	pointer to char	Standard
time_of_launch	int	Standard
Calls		
Function	**Where Described**	
create_dphdr	Sec. 2.1.4.2.3.4	
fio_insert_info	Sec. 2.1.4.2.3.6	
Called By		
Function	**Where Described**	
init_logger_for_copying	Sec. 2.2.2.2.7	
logger_start	Sec. 2.2.2.2.29	

Table 2.1-23: fio_file_header Information.

2.1.4.2.3.4 create_dphdr

This routine combines the passed arguments to form an ASCII string suitable for use as a DataProbe label (reference DataProbe manual). Note that according to the version 8.0 manual, only the following fields are used by DataProbe: *tape_type*, *run_number*, *frame_size*, *header_size*, *time_of_launch*, and *tape_number* (tape files only). This procedure perfoms error checking, and the label is cleared out to the passed length.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
dp_header	pointer to char	Standard
tape_type	pointer to char	Standard
run_number	int	Standard
project_id	pointer to char	Standard
comment	pointer to char	Standard
date_of_run	pointer to char	Standard
tape_number	int	Standard
bits_per_word	int	Standard
frame_size	int	Standard
header_size	int	Standard
date_of_copy	pointer to char	Standard
time_of_run	pointer to char	Standard
time_of_launch	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
field_string[132]	char	Standard
error_string	pointer to char	Standard
lptr	pointer to char	Standard
i	int	Standard
Calls		
Function	**Where Described**	
construct_dphdr	Sec. 2.1.4.2.3.5	
Called By		
Function	**Where Described**	
fio_file_header	Sec. 2.1.4.2.3.3	

Table 2.1-24: create_dphdr Information.

2.1.4.2.3.5 construct_dphdr

This routine adds a field to a DataProbe label. The string to be written, *field_string*, is copied to *field*. If *field_string* is longer than the maximum number of characters allowed in the field, *max_field_length*, it is truncated and a warning message is printed. The item is left justified in its field if the argument *left_justify* is TRUE, and right justified if *left_justify* is FALSE.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
error_string	pointer to char	Standard
max_field_length	int	Standard
field	pointer to char	Standard
field_string	pointer to char	Standard
left_justify	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
string_length	int	Standard
Errors		
Error	**Reason for Error**	
error_string	If *field_string* exceeds *max_field_length*, *field_string* is truncated and a warning message (*field* is too long for DataProbe label) is printed.	
Called By		
Function	**Where Described**	
create_dphdr	Sec. 2.1.4.2.3.4	
set_tape_number_fiohdr	Sec. 2.1.4.2.3.16	

Table 2.1-25: construct_dphdr Information.

2.1.4.2.3.6 *fio_insert_info*

fio_insert_info() inserts a new field specified by *keyword* in the user info buffer of the fio_header specified by *file_hdr*. The user_info field of the fio_header allows the user to store user-defined information in the fio_header. The user info buffer consists of a series of fields each having the following format:

keyword_name char	ifo_bytes short	info... char

The following procedures (Sections 2.1.4.2.3.6-10) are provided to manipulate the user_info portion of the fio_header.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h
keyword	pointer to char	Standard
info	pointer to char	Standard
info_bytes	short	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard
req_bytes	int	Standard
keylen	int	Standard
retval	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
retval	int	if retval = FIO_ERROR, the procedure failed; if retval = FIO_OK, the procedure was successful

Errors	
Error	**Reason for Error**
FIO_HEADER_INFO_BYTES_FULL_ERROR	There is not enough room in the header for the specified information
FIO_HEADER_DUPLICATE_KEYWORD_ERROR	The specified keyword is already in the header.

Calls	
Function	**Where Described**
find_info	Sec. 2.1.4.2.3.8

Called By	
Function	**Where Described**
fio_convert_header	Sec. 2.1.4.2.3.2
fio_file_header	Sec. 2.1.4.2.3.3
set_fileid_fiohdr	Sec. 2.1.4.2.3.11
set_command_fiohdr	Sec. 2.1.4.2.3.13
disk_close	Sec. 2.1.4.2.10 .21

Table 2.1-26: fio_insert_info Information.

2.1.4.2.3.7 info_field_bytes

This routine returns the length in bytes of the info field pointed to by *fldptr*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fldptr	pointer to char	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
info_bytes	short	Standard
keylen	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
keylen + sizeof(short) + info_bytes	int	The length of the info field pointed to by *fldptr*

Called By	
Function	**Where Described**
fio_delete_info	Sec. 2.1.4.2.3.9
find_info	Sec. 2.1.4.2.3.8

Table 2.1-27: info_field_bytes Information.

2.1.4.2.3.8 find_info

This routine finds a field *keyword* in the user info buffer of the fio_header, *file_hdr*, and returns a pointer to it, *ptr*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
keyword	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard
ui_end	pointer to char	Standard
Return Values		
Return Value	**Type**	**Meaning**
ptr	pointer to char	The pointer to the specified field in the user info buffer is returned.
NULL	pointer to char	*The field is null*
Calls		
Function	**Where Described**	
info_field_bytes	Sec. 2.1.4.2.3.7	
Called By		
Function	**Where Described**	
fio_insert_info	Sec. 2.1.4.2.3.6	
fio_delete_info	Sec. 2.1.4.2.3.9	
fio_retrieve_info	Sec. 2.1.4.2.3.10	

Table 2.1-28: find_info Information.

2.1.4.2.3.9 fio_delete_info

This routine deletes the field specified by *keywork* from the user info buffer in *file_hdr*..

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
keyword	pointer to char	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
s	pointer to char	Standard
t	pointer to char	Standard
nbytes	int	Standard
fldbytes	int	Standard
retval	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
retval	int	if retval = FIO_OK, the procedure was successful; if retval = FIO_ERROR, the procedure failed

Errors	
Error	**Reason for Error**
FIO_HEADER_KEYWORD_NOT_FOUND_ ERROR	The specified keyword cannot be found in the header

Calls	
Function	**Where Described**
find_info	Sec. 2.1.4.2.3.8
info_field_bytes	Sec. 2.1.4.2.3.7

Called By	
Function	**Where Described**
set_fileid_fiohdr	Sec. 2.1.4.2.3.11

Table 2.1-29: fio_delete_info Information.

2.1.4.2.3.10 fio_retrieve_info

This routine searches for the field *keyword* in the user info buffer of *file_hdr*, and returns a pointer to it.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
keyword	pointer to char	Standard
info	pointer to pointer to char	Standard
info_bytes	pointer to short	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard
retval	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
retval	int	if retval = FIO_OK, the procedure was successful if retval = FIO_ERROR, the procedure failed.

Errors	
Error	**Reason for Error**
FIO_HEADER_KEYWORD_NOT_FOUND_ERROR	The specified keyword cannot be found in the header

Calls	
Function	**Where Described**
find_info	Sec. 2.1.4.2.3.8

Called By	
Function	**Where Described**
get_fileid_fiohdr	Sec. 2.1.4.2.3.12
get_command_fiohdr	Sec. 2.1.4.2.3.14
fio_print_fiohdr	Sec. 2.1.4.2.3.22
disk_open	Sec. 2.1.4.2.10.5

Table 2.1-30: fio_retrieve_info Information.

The following procedures (Sections 2.1.4.2.3.11-21) are convenience functions for getting and setting standard fields in the fio_header.

2.1.4.2.3.11 set_fileid_fiohdr

This routine sets the *file_id* keyword of the specified fio_header, *file_hdr*. The *file_id* is the name of the data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
file_id	pointer to char	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	The procedure was successful

Errors	
Error	**Reason for Error**
FIO_HEADER_DUPLICATE_KEYWORD_ERROR	The specified keyword is already in the header
FIO_HEADER_KEYWORD_NOT_FOUND_ERROR	The specified keyword cannot be found in the header

Calls	
Function	**Where Described**
fio_insert_info	Sec. 2.1.4.2.3.6
fio_delete_info	Sec. 2.1.4.2.3.9
fio_error	Sec. 2.1.4.2.4.1

Table 2.1-31: set_fileid_fiohdr Information.

2.1.4.2.3.12 get_fileid_fiohdr

This procedure gets the *file_id* keyword field of the specified fio_header, *file_hdr*. The *file_id* is the name of the data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
info	pointer to char	Standard
info_bytes	short	Standard
Return Values		
Return Value	**Type**	**Meaning**
info	pointer to char	The *fileid* keyword field of the specified fio_header
Errors		
Error		**Reason for Error**
FIO_HEADER_KEYWORD_NOT_FOUND_ERROR		The specified keyword cannot be found in the header
Calls		
Function		**Where Described**
fio_retrieve_info		Sec. 2.1.4.2.3.10
Called By		
Function		**Where Described**
fio_print_fiohdr		Sec. 2.1.4.2.3.22
init_logger_for_copying		Sec. 2.2.2.2.7

Table 2.1-32: get_fileid_fiohdr.

2.1.4.2.3.13 set_command_fiohdr

This routine sets the *command* keyword field of the specified fio_header, *file_hdr*. The *command* is the the UNIX command used to create the data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
command	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retv al	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	The procedure was successful
Errors		
Error	**Reason for Error**	
FIO_HEADER_DUPLICATE_KEYWORD_ ERROR	The specified keyword is already in the header	
FIO_HEADER_KEYWORD_NOT_FOUND_ ERROR	The specified keyword cannot be found in the header	
Calls		
Function	**Where Described**	
fio_insert_info	Sec. 2.1.4.2.3.6	
fio_delete_info	Sec. 2.1.4.2.3.9	
fio_error	Sec. 2.1.4.2.4.1	

Table 2.1-33: set_command_fiohdr Information.

2.1.4.2.3.14 get_command_fiohdr

This routine gets the *command* keyword of the specified fio_header, *file_hdr*. The *command* is the the UNIX command used to create the data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
info	pointer to char	Standard
info_bytes	short	Standard
Return Values		
Return Value	**Type**	**Meaning**
info	pointer to char	The COMMAND keyword field of the specified fio_header

Errors	
Error	**Reason for Error**
FIO_HEADER_KEYWORD_NOT_FOUND_ ERROR	The specified keyword cannot be found in the header
Calls	
Function	**Where Described**
fio_retrieve_info	Sec. 2.1.4.2.3.10
Called By	
Function	**Where Described**
fio_print_fiohdr	Sec. 2.1.4.2.3.22

Table 2.1-34: get_command_fiohdr Information.

2.1.4.2.3.15 get_start_date_fiohdr

This procedure returns the start date of the specified fio_header in *date_str*. The *start_date* is the date the exercise was recorded.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
date_str	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dp_ptr	pointer to char	Standard
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	

Table 2.1-35: get_start_date_fiohdr Information.

2.1.4.2.3.16 set_tape_number_fiohdr

This file sets the *tape_number* of the specified fio_header, *file_hdr*. The *tape_number* is the volume number of this file or tape reel within the larger logical data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
tape_number	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dp_ptr	pointer to char	Standard
error_string	pointer to char	Standard
field_string[132]	char	Standard

Calls	
Function	**Where Described**
construct_dphdr	Sec. 2.1.4.2.3.5
Called By	
Function	**Where Described**
write_controller	Sec. 2.1.4.2.10.14
write_controller	Sec. 2.1.4.2.14.28

Table 2.1-36: set_tape_number_fiohdr Information.

2.1.4.2.3.17 get_tape_number_fiohdr

This procedure returns the *tape_number* of the specified fio_header, *file_hdr*. The *tape_number* is the volume number of this file or tape reel within the larger logical data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dp_ptr	pointer to char	Standard
field_string[132]	char	Standard
tape_number	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
tape_number	int	The tape number of the specified header
Called By		
Function		**Where Described**
read_controller		Sec. 2.1.4.2.10.10
disk_open		Sec. 2.1.4.2.10.5
tape_open		Sec. 2.1.4.2.14.16
read_ast_handler		Sec. 2.1.4.2.14.23
tvolume_available		Sec. 2.1.4.2.14.32
tvolume_starttime		Sec. 2.1.4.2.14.33

Table 2.1-37: get_tape_number_fiohdr Information.

2.1.4.2.3.18 set_frame_size_fiohdr

This procedure sets the *frame_size* of the specified fio_header, *file_hdr*. The *frame_size* is the data probe frame size.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
frame_size	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dp_ptr	pointer to char	Standard
error_string	pointer to char	Standard
field_string[132]	char	Standard
Calls		
Function	**Where Described**	
construct_dphdr	Sec. 2.1.4.2.3.5	
Called By		
Function	**Where Described**	
fio_convert_header	Sec. 2.1.4.2.3.2	

Table 2.1-38: set_frame_size_fiohdr Information.

2.1.4.2.3.19 get_start_time_fiohdr

This procedure points to the start_time of the specified fio_header in *time_str*. The *start_date* is the date the exercise was recorded.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
time_str	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dp_ptr	pointer to char	Standard
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	

Table 2.1-39: get_start_time_fiohdr Information.

2.1.4.2.3.20 set_version_chksum_fiohdr

This procedure sets the *version_chksum* of the specified fio_header, *file_hdr*. The *version_chksum* is the checksum identifying which version of the libfio was used to create this data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
version_chksum	int	Standard
Called By		
Function	**Where Described**	
tape_open	Sec. 2.1.4.2.14.16	

Table 2.1-40: set_version_chksum_fiohdr Information.

2.1.4.2.3.21 get_version_chksum_fiohdr

This procedure returns the *fiolib_version_chksum* of the specified fio_header, *file_hdr*. The *version_chksum* is the checksum identifying which version of the libfio was used to create this data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
Return Values		
Return Value	**Type**	**Meaning**
file_hdr-> fiolib_version_chksum	int	The version chksum of the specified fio_header is returned
Called By		
Function	**Where Described**	
disk_open	Sec. 2.1.4.2.10 .5	
read_controller	Sec. 2.1.4.2.10.10	
read_ast_handler	Sec. 2.1.4.2.14.23	
tvolume_available	Sec. 2.1.4.2.14.32	
tvolume_starttime	Sec. 2.1.4.2.14.33	

Table 2.1-41: get_version_chksum_fiohdr Information.

2.1.4.2.3.22 fio_print_fiohdr

This procedure prints the following fields from the specified fio_header, *file_hdr*:

SIMNET data file header info
File Name
Exercise ID's

Project ID
Comment
Date of Run
Time of Run
Volume Number

Command String
Looping mode bof
Fiolib Version

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buffer[DP_HEADER_BYTES]	char	Standard
ptr	pointer to char	Standard
i	int	Standard
size	short	Standard
tapenum	int	Standard
Calls		
Function	**Where Described**	
get_fileid_fiohdr	Sec. 2.1.4.2.3.12	
fio_retrieve_info	Sec. 2.1.4.2.3.10	
get_command_fiohdr	Sec. 2.1.4.2.3.14	
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	
init_logger_for_copying	Sec. 2.2.2.2.7	

Table 2.1-42: fio_print_fiohdr.

2.1.4.2.3.23 null_fiohdr

This procedure sets the specified fio_header, *file_hdr*, to NULL.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_hdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
Called By		
Function	**Where Described**	
init_logger_variables	Sec. 2.2.2.2 logger.c	
disk_open	Sec. 2.1.4.2.10 fiodisk.c	

Table 2.1-43: null_fiohdr.

The file "error.c" includes:
<stdio.h>
"fioerror.h"

External procedure declarations:
perror()
errno()

Variable declarations:
fio_err
err_msgs[NUM_FIO_ERRORS+1]
sys_errlist[]

2.1.4.2.4.1 fio_error

This routine prints out the error message for the last error logged in fio_err.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msg	pointer to char	Standard
Calls		
Function	**Where Described**	
fio_error_string	Sec. 2.1.4.2.4.2	

Called By	
Function	**Where Described**
set_fileid_fiohdr	Sec. 2.1.4.2.3.11
set_command_fiohdr	Sec. 2.1.4.2.3.13
fiofile_records	Sec. 2.1.4.2.10.3
read_dbuffer	Sec. 2.1.4.2.10.9
read_controller	Sec. 2.1.4.2.10.10
write_dbuffer	Sec. 2.1.4.2.10.13
write_controller	Sec. 2.1.4.2.10.14
disk_seek	Sec. 2.1.4.2.10.15
volume_starttime	Sec. 2.1.4.2.10.17
volume_endtime	Sec. 2.1.4.2.10.18
disk_close	Sec. 2.1.4.2.10.21
fnet_open	Sec. 2.1.4.2.10.2
fnet_write	Sec. 2.1.4.2.10.4
eotv_phase_action	Sec. 2.1.4.2.14.3
ini_tbuffers	Sec. 2.1.4.2.14.12
tape_open	Sec. 2.1.4.2.14.16
tape_concat	Sec. 2.1.4.2.14.17
read_fiofile_header	Sec. 2.1.4.2.14.19
read_tbuffer	Sec. 2.1.4.2.14.20
read_stream	Sec. 2.1.4.2.14.21
read_controller	Sec. 2.1.4.2.14.22
read_ast_handler	Sec. 2.1.4.2.14.23
write_fiofile_header	Sec. 2.1.4.2.14.25
write_stream	Sec. 2.1.4.2.14.27
write_controller	Sec. 2.1.4.2.14.28
write_ast_handler	Sec. 2.1.4.2.14.29
tape_seek	Sec. 2.1.4.2.14.30
tvolume_available	Sec. 2.1.4.2.14.32
tvolume_starttime	Sec. 2.1.4.2.14.33
backup_past_target	Sec. 2.1.4.2.14.34
logger_init_cmc	Sec. 2.2.2.2.11
logger_put_remote_rtc	Sec. 2.2.2.2.14

Table 2.1-44: fio_error Information.

2.1.4.2.4.2 fio_error_string

fio_error_string() returns the error message string for the last error logged in fio_err.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msg	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
msgbuf	array of char	Standard

Return Values		
Return Value	**Type**	**Meaning**
msgbuf	char	The error message string in error_msg[fio_err]

Called By	
Function	**Where Described**
open_error_string	Sec. 2.2.2.2 logger.c (macro definition)
logger_playback_state	Sec. 2.2.2.2.18
logger_record_state	Sec. 2.2.2.2.19
fio_error	Sec. 2.1.4.2.4.1

Table 2.1-45: fio_error_string Information.

2.1.4.2.5 fiocvt.h

This file declares the procedure, **cvt_fio_packet()**, defined in "fiocvt.c".

2.1.4.2.6 fiocvt.c

This file is used in converting files recorded using old versions of libfio.

Includes:
<stdio.h>
<errno.h>
<sys/types.h>
"network.h"
"fiolib.h"
"fiofile.h"

Typedefs:

FIO_PACKET_HEADER_V2	fio_packet_header_v2
FIO_PACKET_V2	fio_packet_v2

Define:
FIO_PACKET_SIZE_V2

2.1.4.2.6.1 cvt_fio_packet

This procedure converts an input fio_packet, *inpkt*, from the Version2 to the latest format and points to the packet in *outpkt*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
flags	int	Standard
outpkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
inpkt	pointer to char	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pkt	pointer to fio_packet_v2	Sec. 2.1.4.2.6 fiocvt.c

Calls	
Function	**Where Described**
IS_VERSION	Sec. 2.1.4.1.1 fiolib.h (macro definition)
COPY_FIO_PACKET	Sec. 2.1.4.1.1 fiolib.h (macro definition)
Called By	
Function	**Where Described**
disk_read	Sec. 2.1.4.2.10.7
tape_concat	Sec. 2.1.4.2.14.17

Table 2.1-46: cvt_fio_pkt Information.

2.1.4.2.6.2 fio_packet_size

This procedure returns the fio_packet_size of the specified fio_packet, *pkt*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
flags	int	Standard
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	retval equals the size of the fio_packet
Errors		
Error	**Reason for Error**	
MAX_FIO_PACKET_SIZE	The fio_packet exceeds the maximum allowable size	
Calls		
Function	**Where Described**	
IS_VERSION1	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
IS_VERSION2	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
FIO_PACKET_SIZE_V2	Sec. 2.1.4.2.6 fiocvt.c (macro definition)	
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)	

Table 2.1-47: fio_packet_size Information.

2.1.4.2.7 fioutil.h

This file declares the following procedures defined in "fioutil.c":

fio_alloc()
fio_free()

2.1.4.2.8 fioutil.c

These files contain low level utilities.

Standard procedure declarations:
malloc()
free()

2.1.4.2.8.1 fio_alloc

This procedure allocates and initializes to zero a block of memory by using Standard procedure **malloc()**.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num	unsigned	Standard
size	unsigned	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard
Return Values		
Return Value	**Type**	**Meaning**
ptr	pointer to char	ptr points to the allocated block of memory
Called By		
Function	**Where Described**	
disk_open	Sec. 2.1.4.2.10.5	
fnet_open	Sec. 2.1.4.2.12.2	
new_tbq	Sec. 2.1.4.2.14.7	
ini_tbuffers	Sec. 2.1.4.2.14.12	
tape_open	Sec. 2.1.4.2.14.16	

Table 2.1-48: fio_alloc Information.

2.1.4.2.8.2 fio_free

This procedure frees a block of memory using Standard procedure **free()**.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard

Called By	
Function	**Where Described**
disk_close	Sec. 2.1.4.2.10.21
fnet_open	Sec. 2.1.4.2.12.2
tape_open	Sec. 2.1.4.2.14.16
tape_close	Sec. 2.1.4.2.14.36
write_controller	Sec. 2.1.4.2.14.28

Table 2.1-49: fio_free Information.

2.1.4.2.9 fiodisk.h

This file declares the procedural entry points for **ini_disk()**, **disk_open()**, **disk_concat()**, **disk_read()**, **disk_write()**, **disk_seek()**, and **disk_close()**.

2.1.4.2.10 fiodisk.c

The file "fiodisk.c" contains the procedures used to handle disk based SIMNET data files.

Includes:

Standard libraries:

<stdio.h>
<errno.h>
<fcntl.h>
<unistd.h>
<sys/types.h>
<sys/stat.h>
<ustat.h>

fio files:

"fiolib.h"
"fiocvt.h"
"fioerror.h"
"fiofile.h"
"fioheader.h"
"fioutil.h"
"fiodisk.h"

Constant and Macro defines:

MAX_DISK_VOLUMES
DISK_MODE
MAX_FIO_RECORDS
DISKINFO_PTR()
CONTIGUOUS()
VOLUME_UFD()
READ_BYTES()

Typedefs:

LINT_T	lint_t
FIO_DISK_FILE	disk_t

External procedure declarations:
lseek()

Static procedure declarations:
next_disk_volume()
open_UNIX_disk_file()
fiofile_records()
fio_record_endtime()
read_dp_dbuffer()
read_dbuffer()
read_controller()
write_dp_dbuffer()
write_dbuffer()
write_controller()

The routines for writing input fio_packets from fio_files are called from the "fiolib.c" routine **fio_write()**. **fio_write()** calls the disk routine **disk_write()** to write a single data packet into a buffer. When the buffer is filled, the routine **write_dbuffer()** is called to write the full buffer of fio_packets to the specified disk-based data file. If the disk file is full, the routine **write_controller()** is called to change the current disk file volume.

The routines for reading input fio_packets from fio_files are called from the "fiolib.c" routine **fio_read()**. **fio_read()** calls the disk routine **disk_read()**. **disk_read()** reads a single data packet from the current buffer. When the disk buffer is empty, the routine **read_dbuf()** is called to read another buffer of fio_packets. When the end of volume is reached, the routine **read_controller()** is called to change the current disk file volume.

2.1.4.2.10.1 next_disk_volume

This procedure returns a TRUE if there is a next volume for the disk file *fdptr* and a FALSE if the current volume is the last volume. The index of the next disk volume is pointed to by *ufd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
ufd	pointer to int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Standard

Return Values		
Return Value	**Type**	**Meaning**
TRUE	BOOL	There is a next volume for this disk file
FALSE	BOOL	This is the last volume for this disk file

Calls	
Function	**Where Described**
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
IS_LOOPING	Sec. 2.1.4.1.1 fiolib.h (macro definition)

Called By	
Function	**Where Described**
read_controller	Sec. 2.1.4.2.10.10
write_controller	Sec. 2.1.4.2.10.14

Table 2.1-50: next_disk_volume Information.

2.1.4.2.10.2 open_UNIX_disk_file

This routine opens the UNIX disk file specified in *path* and returns its UNIX file descriptor in *ufd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
path	pointer to char	Standard
fio_flags	int	Standard
size	unsigned long	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
UNIX_flags	int	Standard
ufd	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
ufd	int	The file is opened properly according to its fio_flags and UNIX_flags designations
Called By		
Function	**Where Described**	
disk_concat	Sec. 2.1.4.2.10.6	
disk_open	Sec. 2.1.4.2.10.5	

Table 2.1-51: open_UNIX_disk_volume Information.

2.1.4.2.10.3 fiofile_records

This routine returns the number of fio_records for the file specified in *path*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
path	pointer to char	Standard
is_contiguous	BOOL	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ustat_buf	struct ustat	Standard
stat_buf	struct stat	Standard
blocks	int	Standard
retval	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
retval	*int*	*The number of fio_records for this file. If this value exceeds the max value, an error is returned.*
Errors		
Error	**Reason for Error**	
FIO_WARNING ERROR	fiolib error: exceeded the max number of records for this file.	
Calls		
Function	**Where Described**	
fio_error	Sec. 2.1.4.2.2.2	
Called By		
Function	**Where Described**	
disk_open	Sec. 2.1.4.2.10.5	
disk_concat	Sec. 2.1.4.2.10.6	

Table 2.1-52: fiofile_records Information.

2.1.4.2.10.4 ini_disk

This routine initializes the disk package. This function does not take any parameters, does not return any values, does not make any calls, and does not generate any errors.

Called By	
Function	**Where Described**
ini_fiolib	Sec. 2.1.4.2.2.13

Table 2.1-53: ini_disk Information.

2.1.4.2.10.5 disk_open

This routine opens a disk file for reading or writing. **disk_open()** is able to handle file flags for previous versions of the fio_file data format.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
path	pointer to char	Standard
size	unsigned long	Standard
header	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to register disk_t	Sec. 2.1.4.2.10 fiodisk.c
ufd	int	Standard
curr_fd	int	Standard
retval	int	Standard
ptr	pointer to char	Standard
curloc	long	Standard
bytes	short	Standard
space	lint_t	Sec. 2.1.4.2.10 fiodisk.c

Return Values		
Return Value	**Type**	**Meaning**
retval	int	if retval = FIO_OK, the procedure was successful; if retval = disk_close(fdptr), the procedure has failed and closed the disk file
FIO_OPEN_UNIX_ERROR	int	procedure failed and disk file is closed
FIO_ALLOC_UNIX_ERROR	int	procedure failed and disk file is closed
FIO_VERSION2_ERROR	int	procedure failed and disk file is closed

Errors	
Error	**Reason for Error**
FIO_OPEN_UNIX_ERROR	Unable to open specified file due to a UNIX error.
FIO_ALLOC_UNIX_ERROR	Unable to allocate space for fiolib buffers.
FIO_WARNING_ERROR	Unable to lock in disk buffer.
FIO_VERSION1_ERROR	This fio file volume has an invalid version ID (Version 1)
FIO_VERSION2_ERROR	This fio volume has an invalid version ID (Version 2)
FIO_SEEK_UNIX_ERROR	Disk_open: Seek to logical_bolf

Calls	
Function	**Where Described**
open_UNIX_disk_file	Sec. 2.1.4.2.10.2
fio_alloc	Sec. 2.1.4.2.8.1
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
fiofile_records	Sec. 2.1.4.2.10.3
CONTIGUOUS	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
disk_close	Sec. 2.1.4.2.10.21
IS_READABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
IS_VERSION1	Sec. 2.1.4.1.1 fiolib.h (macro definition)
IS_VERSION2	Sec. 2.1.4.1.1 fiolib.h (macro definition)
read_fiofile_header	Sec. 2.1.4.2.10.8
COPY_FIOHDR	Sec. 2.1.4.1.3 fioheader.h (macro definition)
get_version_chksum_fiohdr	Sec. 2.1.4.2.2 .21
fio_retrieve_info	Sec. 2.1.4.2.3.10
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
null_fiohdr	Sec. 2.1.4.2.3.23
read_dbuffer	Sec. 2.1.4.2.10.9
write_fiofile_header	Sec. 2.1.4.2.10.12
FIO_RECORD_PACKETS	Sec. 2.1.4.1.1 fiofile.h (macro definition)
get_tape_number_fiohdr	Sec. 2.1.4.2.3.17

Table 2.1-54: disk_open Information.

2.1.4.2.10.6 disk_concat

This routine concatenates the disk file specified by *path* to the previously opened disk file specified by *fdptr*. The procedure checks for consistency; disk files may only be concatenated to other disk files. This procedure is able to handle file flags for previous versions of the fio_file data format.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
path	pointer to char	Standard
size	unsigned long	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to register disk_t	Sec. 2.1.4.2.10 fiodisk.c
vol	int	Standard
ufd	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	Procedure was successful
FIO_LOOPING_CONCAT_ ERROR	int	Procedure failed; disk files cannot be concatenated when in LOOPING mode
FIO_MAX_VOLUMES_ ERROR	int	Procedure failed; maximum number of file volumes exceeded
FIO_OPEN_UNIX_ERROR	int	Procedure failed due to a UNIX OPEN error

Errors	
Error	**Reason for Error**
FIO_OPEN_UNIX_ERROR	Unable to open specified file due to a UNIX error.
FIO_ALLOC_UNIX_ERROR	Unable to allocate space for fiolib buffers.
FIO_WARNING_ERROR	Unable to lock in disk buffer.
FIO_VERSION1_ERROR	This fio file volume has an invalid version ID (Version 1)
FIO_VERSION2_ERROR	This fio volume has an invalid version ID (Version 2)
FIO_SEEK_UNIX_ERROR	Seek to logical_bolf error

Calls	
Function	**Where Described**
IS_LOOPING	Sec. 2.1.4.1.1 fiolib.h (macro definition)
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
open_UNIX_disk_file	Sec. 2.1.4.2.10.2
fiofile_records	Sec. 2.1.4.2.10.3

Called By	
Function	**Where Described**
fio_concat	Sec. 2.1.4.2.2.6

Table 2.1-55: disk_concat Information.

2.1.4.2.10.7 disk_read

disk_read reads one data packet from the specified fio_file. The user can get either a pointer to the packet or a copy of the packet by using the *copy* argument. If the proper open flags have been used, the packet will be converted to the latest format before being returned. This routine is called from the "fiolib.c" routine **fio_read**.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
pkt	pointer to pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
copy	BOOL	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to register disk_t	Sec. 2.1.4.2.10 fiodisk.c
status	int	Standard
cvt_buffer [MAX_FIO_PACKET_SIZE]	static char	Standard

Return Values		
Return Value	**Type**	**Meaning**
status	int	if status = FIO_OK, the procedure was successful; if status = an error message, the procedure failed

Calls	
Function	**Where Described**
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
read_dbuffer	Sec. 2.1.4.2.10.9
IS_VERSION	Sec. 2.1.4.1.1 fiolib.h (macro definition)
cvt_fio_packet	Sec. 2.1.4.2.6.1
FIO_PACKET_PTR	Sec. 2.1.4.2.2 fiolib.c (macro definition)

Called By	
Function	**Where Described**
fio_read	Sec. 2.1.4.2.2.7

Table 2.1-56: disk_read Information.

2.1.4.2.10.8 read_fiofile_header

read_fiofile_header reads the fiofile_header from the specified fio file. This procedure is able to handle file flags for previous versions of the fio_file data format.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to register disk_t	Sec. 2.1.4.2.10 fiodisk.c
ufd	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	Procedure was successful
FIO_VERSION1_ERROR	int	Procedure failed

Errors	
Error	**Reason for Error**
FIO_VERSION1_ERROR	This fio file volume has an invalid version ID

Calls	
Function	**Where Described**
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
Called By	
Function	**Where Described**
disk_open	Sec. 2.1.4.2.10.5
read_controller	Sec. 2.1.4.2.10.10

Table 2.1-57: read_fiofile_header Information.

2.1.4.2.10.9 read_dbuffer

read_dbuffer reads a full fio_record of fio_packets into a disk buffer. This procedure is able to handle file flags for previous versions of the fio_file data format. This routine is called by **disk_read()** when the current disk buffer is exhausted.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to register disk_t	Sec. 2.1.4.2.10 fiodisk.c
ufd	int	Standard
nbytes	int	Standard
bptr	pointer to char	Standard
curloc	long	Standard
retval	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	Procedure was successful.
FIO_VERSION1_ERROR	int	Procedure failed.
Errors		
Error	**Reason for Error**	
FIO_EOF_ERROR	End of file reached.	
Calls		
Function	**Where Described**	
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
IS_LOOPING	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
FIO_RECORD_PACKETS	Sec. 2.1.4.1.1 fiofile.h (macro definition)	
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.1.1 fiofile.h (macro definition)	
read_controller	Sec. 2.1.4.2.10.10	
fio_error	Sec. 2.1.4.2.4.1	

Called By	
Function	**Where Described**
disk_open	Sec. 2.1.4.2.10.5
disk_read	Sec. 2.1.4.2.10.7

Table 2.1-58: read_dbuffer Information.

2.1.4.2.10.10 read_controller

read_controller changes the current disk file volume. This procedure is able to handle file flags for previous versions of the fio_file data format. This routine is called by **read_dbuffer()** when the current volume is depleted.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to register disk_t	Sec. 2.1.4.2.10 fiodisk.c
header	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
dfd	int	Standard
ufd	int	Standard
volume_number	int	Standard
curloc	long	Standard
retval	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	Procedure was successful.
retval	int	if retval = 0, the procedure was successful; if retval = an error code, the procedure failed

Errors	
Error	**Reason for Error**
FIO_SEEK_UNIX_ERROR	Read controller: seek to first dbuffer
FIO_VERSION2_ERROR	Ignoring tape volume with invalid format
FIO_VERSION1_ERROR	Ignoring tape volume with invalid format
FIO_WARNING_ERROR	Volume is out of sequence
FIO_EOV_ERROR	End of Volume
FIO_EOF_ERROR	End of File

Calls	
Function	**Where Described**
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
IS_LOOPING	Sec. 2.1.4.1.1 fiolib.h (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
next_disk_volume	Sec. 2.1.4.2.10.1
IS_VERSION1	Sec. 2.1.4.1.1 fiolib.h (macro definition)
read_fiofile_header	Sec. 2.1.4.2.10.8
get_version_chksum_fiohdr	Sec. 2.1.4.2.3.21
IS_VERSION2	Sec. 2.1.4.1.1 fiolib.h (macro definition)
get_tape_number_fiohdr	Sec. 2.1.4.2.3.17
fio_error	Sec. 2.1.4.2.4.1
Called By	
Function	**Where Described**
read_dbuffer	Sec. 2.1.4.2.10.9

Table 2.1-59: read_controller Information.

2.1.4.2.10.11 disk_write

disk_write writes an fio_packet, *pkt*, from a specified fio_file, to a buffer, *dptr*. This routine calls **write_dbuffer**(). This routine is called from the rest of the DataLogger code through the "libfio.c" routine **fio_write**().

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
status	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
status	int	if status = FIO_OK, the procedure was successful; if status = an error code, the procedure failed
Calls		
Function	**Where Described**	
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
write_dbuffer	Sec. 2.1.4.2.10.13	
FIO_RECORD_PACKETS	Sec. 2.1.4.1.1 fiofile.h (macro definition)	
COPY_FIO_PACKET	Sec. 2.1.4.1.1 fiolib.h (macro definition)	

Called By	
Function	**Where Described**
fio_write	Sec. 2.1.4.2.2.8

Table 2.1-60: disk_write Information.

2.1.4.2.10.12 write_fiofile_header

This routine writes out the fiofile headerto the specified data file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
ufd	int	Standard
hbytes	unsigned	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	Procedure was successful
FIO_WRITE_UNIX_ERROR	int	Procedure failed
Errors		
Error	**Reason for Error**	
FIO_WRITE_UNIX_ERROR	Write failed due to UNIX error.	
Calls		
Function	**Where Described**	
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
CONTIGUOUS	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
Called By		
Function	**Where Described**	
write_controller	Sec. 2.1.4.2.10.14	
disk_close	Sec. 2.1.4.2.10.21	
disk_open	Sec. 2.1.4.2.10.5	

Table 2.1-61: write_fiofile_header Information.

2.1.4.2.10.13 write_dbuffer

This routine writes a full disk buffer of fio_packets to a disk data file. This routine is called by **disk_write**() when the current disk buffer is full.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
ufd	int	Standard
nbytes	int	Standard
retval	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	If retval = FIO_OK, the procedure was successful; If retval = an error code, the procedure failed
Errors		
Error	**Reason for Error**	
EFBIG	UNIX error	
FIO_INTERNAL_ERROR	Premature end of contiguous file	
FIO_EOF_ERROR	End of file	
FIO_WRITE_UNIX_ERROR	Write failed due to UNIX error	
Calls		
Function	**Where Described**	
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
FIO_RECORD_PACKETS	Sec. 2.1.4.1.1 fiofile.h (macro definition)	
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.1.1 fiofile.h (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
write_controller	Sec. 2.1.4.2.10.14	
CONTIGUOUS	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
fio_error	Sec. 2.1.4.2.4.1	
Called By		
Function	**Where Described**	
disk_write	Sec. 2.1.4.2.10.11	
disk_close	Sec. 2.1.4.2.10.21	

Table 2.1-62: write_dbuffer Information.

2.1.4.2.10.14 write_controller

write_controller() changes the current disk file volume. This routine is called by **write_dbuffer**() when the current volume is full.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
dfd	int	Standard
ufd	int	Standard
curloc	long	Standard
retval	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	If retval = FIO_OK, the procedure was successful; If retval = an error code, the procedure failed
FIO_EOV_ERROR	int	procedure failed

Errors	
Error	**Reason for Error**
FIO_SEEK_UNIX_ERROR	Write controller: seek to first dbuffer
FIO_EOV_ERROR	End of volume
FIO_EOF_ERROR	End of file

Calls	
Function	**Where Described**
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
IS_LOOPING	Sec. 2.1.4.1.1 fiolib.h (macro definition)
fio_error	Sec. 2.1.4.2.4.1
next_disk_volume	Sec. 2.1.4.2.10.1
set_tape_number_fiohdr	Sec. 2.1.4.2.3.16
write_fiofile_header	Sec. 2.1.4.2.10.12

Called By	
Function	**Where Described**
wirte_dbuffer	Sec. 2.1.4.2.10.13

Table 2.1-63: write_controller Information.

2.1.4.2.10.15 disk_seek

disk_seek seeks to a specified location in a disk file. First the time *offset* parameter is translated into an absolute time to seek. Then the routine checks to see if the seek time is outside the beginning or end of the file. If the seek time is within the fio_file, the file pointer, *ufd*, is set to the correct disk volume. Then the seek tries to find the correct packet, returning FIO_OK if successful or an error message if unsuccessful. This routine interfaces with the rest of the DataLogger code through the "libfio.c" routine **fio_seek**().

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
offset	int	Standard
whence	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
ptr	pointer to char	Standard
buffer [READ_BYTES]	char	Standard
ofs	long	Standard
i	int	Standard
ufd	int	Standard
retval	int	Standard
status	int	Standard
abstime	int	Standard
curtime	int	Standard
starttime	int	Standard
endtime	int	Standard
distance	long	Standard
remainder	long	Standard

Return Values		
Return Value	**Type**	**Meaning**
status	int	procedure failed due to a starttime or endtime error
retval	int	procedure was successful (retval equals FIO_OK)
FIO_SEEK_INVALID_ WHENCE_VALUE_ERROR	int	procedure failed due to an invalid *whence* parameter
FIO_READ_UNIX_ERROR	int	procedure failed due to a UNIX READ error
FIO_SEEK_UNIX_ERROR	int	procedure failed due to a UNIX SEEK error

Errors	
Error	**Reason for Error**
FIO_SEEK_UNIX_ERROR	Seek failed due to UNIX error
FIO_SEEK_INVALID_WHENCE_VALUE_ ERROR	Seek called with an invalid parameter for the *whence* parameter
FIO_SEEK_PAST_BOF_ERROR	Attempting to seek past the beginning of the specified file
FIO_SEEK_PAST_EOF_ERROR	Attempting to seek past the end of the specified file
FIO_INTERNAL_ERROR	- Distance of seek to file end is 0 - Curvol < 0 - Ran off the end of the dbuffer without packet.
FIO_READ_UNIX_ERROR	Read failed due to UNIX error

Calls	
Function	**Where Described**
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)
fio_record_endtime	Sec. 2.1.4.2.10.19
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)
volume_starttime	Sec. 2.1.4.2.10.17
volume_endtime	Sec. 2.1.4.2.10.18
fio_error	Sec. 2.1.4.2.4.1
FIO_RECORD_PACKETS	Sec. 2.1.4.1.1 fiofile.h (macro definition)
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.1.1 fiofile.h (macro definition)
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
Called By	
Function	**Where Described**
fio_seek	Sec. 2.1.4.2.2.9

Table 2.1-64: disk_seek Information.

2.1.4.2.10.16 disk_seek_time

disk_seek_time returns the time stamp of the current fio_packet.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
Return Values		
Return Value	**Type**	**Meaning**
FIO_PACKET_TIME	int	the time stamp of the specified fio_packet
0	int	the procedure failed
Calls		
Function		**Where Described**
DISKINFO_PTR		Sec. 2.1.4.2.10 fiodisk.c (macro definition)
FIO_PACKET_TIME		Sec. 2.1.4.1.1 fiolib.h (macro definition)
Called By		
Function		**Where Described**
fio_seek		Sec. 2.1.4.2.2.9

Table 2.1-65: disk_seek_time Information.

2.1.4.2.10.17 volume_starttime

This routine sets a pointer, *time*, to the timestamp of the first fio_packet in the file volume specified in *ufd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ufd	int	Standard
time	pointer to int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	Standard
buffer[READ_BYTES]	char	Standard
ofs	long	Standard
oldloc	long	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	procedure was sucessful
FIO_SEEK_UNIX_ERROR	int	procedure failed due to UNIX SEEK error
FIO_READ_UNIX_ERROR	int	procedure failed due to UNIX READ error
Errors		
Error	**Reason for Error**	
FIO_SEEK_UNIX_ERROR	Procedure failed due to UNIX error	
FIO_INTERNAL_ERROR	Volume start time < 0	
FIO_READ_UNIX_ERROR	Procedure failed due to UNIX error	
Calls		
Function	**Where Described**	
FIO_RECORD_PACKETS	Sec. 2.1.4.1.1 fiofile.h (macro definition)	
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
fio_error	Sec. 2.1.4.2.4.1	
Called By		
Function	**Where Described**	
disk_seek	Sec. 2.1.4.2.10 .15	

Table 2.1-66: volume_starttime Information.

2.1.4.2.10.18 volume_endtime

This routine sets a pointer, *time*, to the timestamp of the last fio_packet in the file volume specified in *ufd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ufd	int	Standard
time	pointer to int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
buffer[FIO_RECORD_BYTES]	char	Standard
frcd	pointer to fio_record	Sec. 2.1.4.1.1 fiofile.h
ofs	long	Standard
oldloc	long	Standard
newloc	long	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	procedure was successful
FIO_SEEK_UNIX_ERROR	int	procedure failed due to UNIX SEEK error
FIO_READ_UNIX_ERROR	int	procedure failed due to UNIX READ error

Errors	
Error	**Reason for Error**
FIO_SEEK_UNIX_ERROR	Procedure failed due to UNIX error
FIO_INTERNAL_ERROR	Volume end time < 0
FIO_READ_UNIX_ERROR	Procedure failed due to UNIX error

Calls	
Function	**Where Described**
fio_error	Sec. 2.1.4.2.4.1

Called By	
Function	**Where Described**
disk_seek	Sec. 2.1.4.2.10.15

Table 2.1-67: volume_endtime Information.

2.1.4.2.10.19 fio_record_endtime

This procedure sets a pointer, *time*, to the last fio_packet in the fio_record specified in *frcd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
frcd	pointer to fio_record	Sec. 2.1.4.1.1 fiofile.h
time	pointer to int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
current_packet	pointer to char	Standard
eo_packets	pointer to char	Standard
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h

Calls	
Function	**Where Described**
FIO_RECORD_PACKETS	Sec. 2.1.4.1.1 fiofile.h (macro definition)
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.1.1 fiofile.h (macro definition)
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)
Called By	
Function	**Where Described**
disk_seek	Sec. 2.1.4.2.10.15

Table 2.1-68: fio_record_endtime Information.

2.1.4.2.10.20 disk_getrawfd

This routine returns the raw UNIX file descriptor for the current volume specified in *fdptr*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
vol_offset	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
tvol	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
dptr->volume_ufds[tvol]	pointer to disk_t	The raw UNIX file descriptor
Calls		
Function	**Where Described**	
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
Called By		
Function	**Where Described**	
fio_getrawfd	Sec. 2.1.4.2.2.10	
fio_flush	Sec. 2.1.4.2.2.11	

Table 2.1-69: disk_getrawfd Information.

2.1.4.2.10.21 disk_close

This routine closes the specified disk file. The disk buffer is freed, the FIO_DISK_FILE is freed, and the header is freed. This routine is called from the rest of the libfio code through the fiolib.c routine **fio_close()**.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.1.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
dptr	pointer to disk_t	Sec. 2.1.4.2.10 fiodisk.c
ufd	int	Standard
i	int	Standard
offset	long	Standard
retval=FIO_OK	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	procedure was successful
FIO_SEEK_UNIX_ERROR	int	procedure failed due to UNIX SEEK error
FIO_READ_UNIX_ERROR	int	procedure failed due to UNIX READ error
Errors		
Error	**Reason for Error**	
FIO_INTERNAL_ERROR	- No room for LOOPING pointer - Write of header failed	
FIO_CLOSE_UNIX_ERROR	UNIX CLOSE error	
Calls		
Function	**Where Described**	
DISKINFO_PTR	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.10 fiodisk.c (macro definition)	
IS_WRITABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
write_dbuffer	Sec. 2.1.4.2.10.13	
IS_LOOPING	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
fio_insert_info	Sec. 2.1.4.2.3.6	
fio_error	Sec. 2.1.4.2.4.1	
write_fiofile_header	Sec. 2.1.4.2.10.8	
fio_free	Sec. 2.1.4.2.8.2	
Called By		
Function	**Where Described**	
fio_close	Sec. 2.1.4.2.2.12	
disk_open	Sec. 2.1.4.2.10.5	

Table 2.1-70: disk_close Information.

2.1.4.2.11 fionet.h

This file declares the procedural entry points for **ini_fnet()**, **fnet_open()**, **fnet_read()**, **fnet_write()**, **fnet_flush()**, and **fnet_close()**.

2.1.4.2.12 fionet.c

The file "fionet.c" contains the procedures used to handle network based SIMNET data files.

Includes:
- standard library:
  - <stdio.h>
- fio files:
  - "fiolib.h"
  - "fioerror.h"
  - "fiofile.h"
  - "fioheader.h"
  - "fioutil.h"
  - "fionet.h"

Typedefs:

LINT_T	lint_t
FIO_NET_FILE	fnet_t

Static Variable Declarations:
- errmsg

Constant and Macro Defines:
- PACKETS_PER_LATE_CHECK
- LATE_MARGIN
- FNETINFO_PTR

This file provides the routines for SIMNET (CMC) Network based fio_packet input/output operations. No attempt is made to buffer fio_packets being read or written to a network file. Note that it may be undesirable for an application built on the fiolib to BLOCK on reads and writes. As a result, NON_BLOCKING mode can be specified for network files.

2.1.4.2.12.1 ini_fnet

This procedure initializes the fnet package. This procedure takes no parameters, returns no values, and has no errors or internal parameters. **ini_fnet()** is called from the "fiolib.c" function **ini_fiolib()**.

Called By	
Function	**Where Described**
ini_fiolib	Sec. 2.1.4.2.2 fiolib.c

Table 2.1-71: ini_fnet Information.

2.1.4.2.12.2 fnet_open

This routine opens and initializes the specified network file. It initializes the network in the appropriate modes.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
path	pointer to char	Standard
header	pointer to fio_header	Sec. 2.1.4.1.1 fioheader.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
nptr	pointer to fnet_t	Sec. 2.1.4.2.12 fionet.c
nfd	int	Standard
scrap_net_address	int	Standard
space	lint_t	Sec. 2.1.4.2.12 fionet.c

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	procedure was successful
FIO_ALLOC_UNIX_ERROR	int	procedure failure due to inability to allocate space for fiolib buffers
FIO_NET_OPEN_ERROR	int	procedure failure due to inablility to open network device card

Errors	
Error	**Reason for Error**
FIO_NET_OPEN_ERROR	Unable to open network device card
FIO_ALLOC_UNIX_ERROR	Unable to allocate space for fiolib buffers
FIO_WARNING_ERROR	Unable to lockin network file buffer
FIO_NET_PROMISCUOUS_MODE_ERROR	Unable to activate promiscuous mode
FIO_NET_NORMAL_MODE_ERROR	Unable to activate normal mode
FIO_NET_ALIVE_ERROR	Network device is not alive
FIO_NET_STAMP_ENABLE_ERROR	Unable to enable timestamping on network device
FIO_NET_SETTIME_ERROR	Unable to reset card statistics on network device
FIO_NET_FLUSH_ERROR	Unable to flush network device rings
FIO_NET_ZERO_STATISTICS_ERROR	Unable to zero CMC card time

Calls	
Function	**Where Described**
net_open	Sec. 2.20.2 in MCC CSCI SDD
fio_alloc	Sec. 2.1.4.2.8.1
fio_error	Sec. 2.1.4.2.4.1
FNETINFO_PTR	Sec. 2.1.4.2.12 fionet.c (macro definition)
FIO_PACKET_PTR	Sec. 2.1.4.1.1 fiolib.h (macro definition)
net_stop	Sec. 2.20.2 in MCC CSCI SDD
net_prom	Sec. 2.20.2 in MCC CSCI SDD
net_norm	Sec. 2.20.2 in MCC CSCI SDD
net_alive	Sec. 2.20.2 in MCC CSCI SDD
net_init_mca	Sec. 2.20.2 in MCC CSCI SDD
net_stamp_enable	Sec. 2.20.2 in MCC CSCI SDD
net_settime	Sec. 2.20.2 in MCC CSCI SDD
net_flush	Sec. 2.20.2 in MCC CSCI SDD

Table 2.1-72: fnet_open Information.

2.1.4.2.12.3 fnet_read

The routine reads a single PDU from the network into an fio_packet along with addressing, timestamp, and type information. It passes back either a pointer to this fio_packet or a copy, depending on the *copy* argument. This routine is called from the rest of the DataLogger code through **fio_read()** in the file "fiolib.c".

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
pkt	pointer to pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
copy	BOOL	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pptr	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
nptr	pointer to fnet_t	Sec. 2.1.4.2.12 fionet.c
nfd	int	Standard
retval	int	Standard
flags	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	If retval = FIO_OK, the procedure was successful; If retval = an error code, the procedure failed
Errors		
Error	**Reason for Error**	
FIO_NO_PACKETS_AVAIL_ERROR	No fio_packets available for reading	

Calls	
Function	**Where Described**
FNETINFO_PTR	Sec. 2.1.4.2.12 fionet.c (macro definition)
IS_NBLOCKING	Sec. 2.1.4.1.1 fiolib.h (macro definition)
net_rcv	Sec. 2.20.2 in MCC CSCI SDD
net_get_timestamp	Sec. 2.20.2 in MCC CSCI SDD
net_get_rcv_from_addr	Sec. 2.20.2 in MCC CSCI SDD
net_get_rcv_to_addr	Sec. 2.20.2 in MCC CSCI SDD
net_get_rcv_type	Sec. 2.20.2 in MCC CSCI SDD
Called By	
Function	**Where Described**
fio_read	Sec. 2.1.4.2.2.7

Table 2.1-73: fnet_read Information.

2.1.4.2.12.4 fnet_write

This routine gets an fio_packet and writes its PDU out to the network. Normally, this routine call will block until a packet is transmitted, however, if the file is opened using the FIO_NBLOCK flag, the call will return immediately. This routine interfaces with the rest of the DataLogger code through **fio_write()** in the file "fiolib.c".

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
nptr	pointer to fnet_t	Sec. 2.1.4.2.12 fionet.c
nfd	int	Standard
flags	int	Standard
now	int	Standard
retval	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	If retval = FIO_OK, the procedure was successful If retval equals an error message, the procedure failed
Errors		
Error		**Reason for Error**
FIO_INVALID_PACKET_TIMESTAMP_ERROR		Invalid fio_packet timestamp
FIO_INVALID_PACKET_SIZE_ERROR		Invalid fio_packet size
FIO_NET_WRITE_ERROR		Unable to write packet to network device
FIO_LATE_PACKET_TO_NET_ERROR		Late packet to net

Calls	
Function	**Where Described**
FNETINFO_PTR	Sec. 2.1.4.2.12 fionet.c (macro definition)
IS_NBLOCKING	Sec. 2.1.4.1.1 fiolib.h (macro definition)
fio_error	Sec. 2.1.4.2.4.1
net_put_timestamp	Sec. 2.20.2 in MCC CSCI SDD
net_set_snd_from_addr	Sec. 2.20.2 in MCC CSCI SDD
net_set_snd_type	Sec. 2.20.2 in MCC CSCI SDD
net_snd	Sec. 2.20.2 in MCC CSCI SDD
net_gettime	Sec. 2.20.2 in MCC CSCI SDD
Called By	
Function	**Where Described**
fio_write	Sec. 2.1.4.2.2.8

Table 2.1-74: fnet_write Information.

2.1.4.2.12.5 fnet_flush

This routine flushes the network transmit or recive bufter files. This routine interfaces with the rest of the DataLogger code through **fio_flush** in the file "fiolib.c".

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
flags	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
net_flags	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	If retval = FIO_OK, the procedure was successful If retval equals an error message, the procedure failed
Errors		
Error	**Reason for Error**	
FIO_NET_FLUSH_ERROR	Unable to flush network device rings	
Calls		
Function	**Where Described**	
net_flush	Sec. 2.20.2 in MCC CSCI SDD	

Called By	
Function	**Where Described**
fio_flush	Sec. 2.1.4.2.2.11

Table 2.1-75: fnet_flush Information.

2.1.4.2.12.6 fnet_close

This routine closes the network fio_file. This routine interfaces with the rest of the DataLogger code through **fio_close** in the file "fiolib.c".

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard
scrap_net_address	int	Standard
nfd	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	If retval = FIO_OK, the procedure was successful If retval equals an error message, the procedure failed
Errors		
Error		**Reason for Error**
FIO_NET_NORMAL_MODE_ERROR		Unable to activate normal mode
FIO_NET_CLOSE_ERROR		Unable to close network device
Calls		
Function		**Where Described**
net_stop		Sec. 2.20.2 in MCC CSCI SDD
net_norm		Sec. 2.20.2 in MCC CSCI SDD
net_close		Sec. 2.20.2 in MCC CSCI SDD
Called By		
Function		**Where Described**
fio_close		Sec. 2.1.4.2.2.12

Table 2.1-76: fnet_close Information.

2.1.4.2.14 fiotape.h

This file declares the procedural entry points for **ini_tape()**, **tape_open()**, **tape_read()**, **tape_write()**, **tape_seek()**, and **tape_close()**.

2.1.4.2.14 fiotape.c

The key to fast I/O using 9-track tape is to keep the tape drive streaming as much as possible. To do this requires that the tape records be of constant size. Reading from and writing to 9-track tape-based data files requires that these requirements be met.

On the MASSCOMP, streaming is handled using asynchronous I/O via the AST (Asynchronous Software Trap) which is a proprietary extension to UNIX in the MASSCOMP version of UNIX RTU.

The "fiotape.c" routines make use of the AST for writing as follows: individual fio_packets are written into a fio_record buffer using calls to **tape_write**. When the buffer is full it is placed on the end of a list of buffers by a call to **write_buffer**. When enough buffers are full, **write_buffer** calls **write_stream** to start the tape device streaming. This is done asynchronously using IOCTL calls and multiple AST's.

The AST handler **write_ast_handler** checks for error conditions and attempts to keep the drive streaming if there are still full buffers available. Once a particular tape volume (tape reel) has been filled, **write_stream** calls **write_controller** to switch to another volume (tape device, tape reel, or both).

The process for reading from a 9-track tape data file is similar, except that buffers are being emptied instead of filled. Also, the limiting factor on the streaming is the amount of memory available for buffers, not the number of full buffers. In exercises that have very high data rates, having sufficient amounts of internal memory allows the transitions between tape volumes (tape reels) to proceed smoothly.

Includes:

standard libraries:

<stdio.h>
<fcntl.h>
<errno.h>
<ast.h>
<sys/types.h>
<sys/mtio.h>
<sys/ioctl.h>

fio files:

"fiolib.h"
"fiocvt.h"
"fioerror.h"
"fiofile.h"
"fioheader.h"
"fioutil.h"
"fiotape.h"

Constant and Macro Defines:

EOTV_PHASE_DELAY
EOTV_NULL_VOLUME
NULL_TAPE_NUMBER
NULL_STARTTIME
MAX_TAPE_DRIVES
MAX_TAPE_VOLUMES
MAX_VOLUME_STARTTIME
TOO_BIG_2MB_5600
OK_2B_5600_NO_EXCELAN
OK_2MB_5600
MAX_BUFFER_MEMORY
TBUFFERS_PER_600_FOOT_TAPE
TBUFFERS_PER_1200_FOOT_TAPE
TBUFFERS_PER_2400_FOOT_TAPE
TBUFFERS_PER_TINY
TBUFFERS_PER_SMALL
TBUFFERS_PER_VERSION1_TAPE
TAPE_READ_THRESHOLD
MAX_SIMULTANEOUS_OPS
TOTAL_TBUFFERS
FIO_HEADER_OF_TBUFFER
FIO_RECORD_OF_TBUFFER
FIO_RECORD_STARTTIME
VOLUME_NUMBER
VOLUME_STARTTIME
SET_VOLUME_STARTTIME
OPS_AVAILABLE
TBUFS_AVAILABLE
VOLUME_UFD
ZYLOGICS_472_EOT_ERROR
XYLOGICS_472_DRIVE_READY
XYLOGICS_472_DRIVE_ONLINE
TAPE_AST_PRIORITY
DISABLE_AST
RESTORE_AST
WAIT_FOR_AST
MIN
TAPEINFO_PTR
QUEUE_COUNT()
QUEUE_HEAD()
QUEUE_TAIL()

Typedefs:

TBUFFER_Q	tbuffer_q
T_BUFFER_T	tbuffer_t
LINT_T	lint_t
EOTV_E	eotv_e
EOTV_PHASE_E	eotv_phs_e
LLSTATE_E	llstat_e
VOLUME_INFO	tapevol_info
FIO_TAPE_FILE	tape_t

Global Variable Declarations:
errmsg
free_tbq

External procedure declarations:
exit()

Static procedure declarations:
do_eotv_action()
init_eotv_action()
eotv_phase_action()
unix_tape_action()
promote_status_info()
is_next_tape_volume()
read_fiofile_header()
read_tbuffer()
read_stream()
read_controller()
read_ast_handler()
write_fiofile_header()
write_tbuffer()
write_stream()
write_controller()
write_ast_handler()
backup_past_target()
tvolume_starttime()
tvolume_available()

The following routines (Sections 2.1.4.2.14.1-3) handle the transitioning from one tape volume on one tape drive to another tape volume on another tape drive in the case of a concatenated tape file.

2.1.4.2.14.1 init_eotv_action

This routine initializes the end of tape volume action, setting the *volume*, *phase*, and *delay* characteristics from the input parameters.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
volume	int	Standard
phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
delay	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
Calls		
Function		**Where Described**
TAPEINFO_PTR		Sec. 2.1.4.2.14 fiotape.c (macro definition)

Called By	
Function	**Where Described**
read_controller	Sec. 2.1.4.2.14.22
write_controller	Sec. 2.1.4.2.14.28

Table 2.1-77: init_eotv_action Information.

2.1.4.2.14.2 do_eotv_action

This performs the end of tape volume action. The choices are:

1) To do nothing
2) To rewind off-line
3) To rewind on-line

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
delay	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
vol	int	Standard
ufd	int	Standard
phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
action	eotv_e	Sec. 2.1.4.2.14 fiotape.c
Calls		
Function	**Where Described**	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
eotv_phase_action	Sec. 2.1.4.2.14 .3	
Called By		
Function	**Where Described**	
write_tbuffer	Sec. 2.1.4.2.14.26	
read_tbuffer	Sec. 2.1.4.2.14.20	

Table 2.1-78: do_eotv_action Information.

2.1.4.2.14.3 eotv_phase_action

This routine performs the action specified by *eotv_action* on the Unix file descriptor specified in *ufd*. Rewinding the tape is a phased action; since the rewind takes a long time, the action is broken up into phases so the tape streaming can continue. The phases are:

1) Backup a small amount
2) Write the End-of-Tape marker
3) Execute the End-of-Tape action
4) Clear the status

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ufd	int	Standard
eotv_phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
eotv_action	eotv_e	Sec. 2.1.4.2.14 fiotape.c
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
op	struct mtop	Standard
status	struct mtget	Standard
retval	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
Return Values		
Return Value	**Type**	**Meaning**
retval	eotv_ph_e	The End_of_Tape_Volume phase action.
Calls		
Function		**Where Described**
fio_error		Sec. 2.1.4.2.4.1
XYLOGICS_472_DRIVE_READY		Sec. 2.1.4.2.14 fiotape.c (macro definition)
eotv_phase_action		Sec. 2.1.4.2.14.3
Called By		
Function		**Where Described**
tvolume_available		Sec. 2.1.4.2.14.32
do_eotv_action		Sec. 2.1.4.2.14.2
read_controller		Sec. 2.1.4.2.14.22
tape_seek		Sec. 2.1.4.2.14.30
write_controller		Sec. 2.1.4.2.14.28
tvolume_starttime		Sec. 2.1.4.2.14.33

Table 2.1-79: eotv_phase_action Information.

2.1.4.2.14.4 unix_tape_action

This routine performs the specified *action* on the specified Unix file descriptor, *ufd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ufd	int	Standard
action	int	Standard
count	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
op	struct mtop	Standard

Return Values		
Return Value	**Type**	**Meaning**
-1	int	action cannot be performed: outstanding ioctl
0	int	procedure was successful
Called By		
Function	**Where Described**	
backup_past_target	Sec. 2.1.4.2.14.34	

Table 2.1-80: unix_tape_action Information.

2.1.4.2.14.5 promote_status_info

This routine promotes the ll_status to the appropriate hl_status.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c
Calls		
Function	**Where Described**	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
Called By		
Function	**Where Described**	
tape_open	Sec. 2.1.4.2.14.16	

Table 2.1-81: promote_status_info Information.

2.1.4.2.14.6 is_next_tape_volume

This routine returns a pointer to the index of the next tape volume in *vol*. This routine returns a TRUE if there is a next volume for this tape file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
vol	pointer to int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c

Return Values		
Return Value	**Type**	**Meaning**
TRUE	BOOL	there is a next tape volume
FALSE	BOOL	this is the last tape volume
Calls		
Function	**Where Described**	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
IS_CONTINUOUS	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
Called By		
Function	**Where Described**	
write_tbuffer	Sec. 2.1.4.2.14.26	
write_controller	Sec. 2.1.4.2.14.28	
write_ast_handler	Sec. 2.1.4.2.14.29	
read_ast_handler	Sec. 2.1.4.2.14.23	
read_controller	Sec. 2.1.4.2.14.22	
read_stream	Sec. 2.1.4.2.14.21	

Table 2.1-82: is_next_tape_volume Information.

Tape buffers and tape buffer queues

The following routines handle tape buffers so that the tape device can be kept streaming.

2.1.4.2.14.7 new_tbq

This routine allocates and initializes an empty tbuffer_q, *tbq*.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
space	lint_t	Sec. 2.1.4.2.14 fiotape.c
tbq	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c
Return Values		
Return Value	**Type**	**Meaning**
tbq	pointer to tbuffer_q	a pointer to the new tbuffer_q
Calls		
Function	**Where Described**	
fio_alloc	Sec. 2.1.4.2.5.1	
Called By		
Function	**Where Described**	
tape_open	Sec. 2.1.4.2.14.16	
concat_tbs	Sec. 2.1.4.2.14.10	

Table 2.1-83: new_tbq Information.

2.1.4.2.14.8 enqueue_tb

This routine adds a tbuffer, *tbq*, to the end of a tbuffer_q and sets a pointer, *tbuf*, to the number of buffers in the list.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tbq	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c

Called By	
Function	**Where Described**
write_tbuffer	Sec. 2.1.4.2.14.26
write_ast_handler	Sec. 2.1.4.2.14.29
tape_seek	Sec. 2.1.4.2.14.30
tape_concat	Sec. 2.1.4.2.14.17
ini_tbuffer	Sec. 2.1.4.2.14.12
tape_close	Sec. 2.1.4.2.14.36
read_ast_handler	Sec. 2.1.4.2.14.23
tape_open	Sec. 2.1.4.2.14.16

Table 2.1-84: enqueue_tb Information.

2.1.4.2.14.9 dequeue_tb

This routine removes a tbuffer from the head of a tbuffer_q and returns a pointer to the former head of the tbuffer_q.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tbq	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c

Return Values		
Return Value	**Type**	**Meaning**
tbuf	pointer to tbuffer_t	the former head of the tbuffer_q

Called By	
Function	**Where Described**
write_stream	Sec. 2.1.4.2.14.27
next_available_tbuffer	Sec. 2.1.4.2.14.13
tape_seek	Sec. 2.1.4.2.14.30
read_tbuffer	Sec. 2.1.4.2.14.20
tape_close	Sec. 2.1.4.2.14.36
tape_open	Sec. 2.1.4.2.14.16

Table 2.1-85: dequeue_tb Information.

2.1.4.2.14.10 concat_tbqs

This routine creates a new tbuffer_q which is the concatenation of the two specified queues, *tbq1* and *tbq2*, and returns a pointer to the new tape buffer queue.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tbq1	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c
tbq2	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
concat_tbq	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c
Return Values		
Return Value	**Type**	**Meaning**
concat_tbq	pointer to tbuffer_q	a pointer to the new, concatenated tape buffer queue.
Calls		
Function	**Where Described**	
new_tbq	Sec. 2.1.4.2.14.7	
Called By		
Function	**Where Described**	
write_controller	Sec. 2.1.4.2.14.28	

Table 2.1-86: concat_tbs Information.

2.1.4.2.14.11 null_tbq

This routine NULLs out the specified tape buffer queue by setting the *tbq* count field to 0 and the *tbq* head and tail to NULL.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tbq	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c
Called By		
Function	**Where Described**	
write_controller	Sec. 2.1.4.2.14.28	

Table 2.1-87: null_tbq Information.

2.1.4.2.14.12 ini_tbuffers

This routine initializes the tape buffers abstraction. Tape buffers are "locked" into memory for performance reasons. The procedure returns FIO_OK if successful and an error message if unsuccessful.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
initialized	BOOL	Standard
i	int	Standard
space	lint_t	Sec. 2.1.4.2.14 fiotape.c
tbq	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
lockin_failed	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	procedure was successful
FIO_ALLOC_UNIX_ERROR	int	procedure failed due to error
FIO_WARNING_ERROR	int	procedure failed due to error

Errors	
Error	**Reason for Error**
FIO_ALLOC_UNIX_ERROR	Unable to allocate space for fiolib buffers
FIO_WARNING_ERROR	Unable to lockin tape buffers

Calls	
Function	**Where Described**
fio_alloc	Sec. 2.1.4.2.8.1
enqueue_tb	Sec. 2.1.4.2.14.8
fio_error	Sec. 2.1.4.2.4.1

Called By	
Function	**Where Described**
tape_open	Sec. 2.1.4.2.14.16

Table 2.1-88: ini_tbuffers Information.

2.1.4.2.14.13 next_available_tbuffer

This routine gets and initializes an empty tbuffer for the specified fio_file, *fdptr*, from the stash of PRIVATE tbuffers, or if they are exhausted, from the free tbuffer pool. A pointer to an intialized tbuffer, *tbuf*, if returned.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
ast	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
tbuf	pointer to tbuffer_t	the next available initialized tbuffer.
FIO_ALLOC_UNIX_ERROR	int	procedure failed due to error
FIO_WARNING_ERROR	int	procedure failed due to error

Calls	
Function	**Where Described**
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
dequeue_tb	Sec. 2.1.4.2.14.9
IS_READABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)

Called By	
Function	**Where Described**
tape_write	Sec. 2.1.4.2.14.24
read_stream	Sec. 2.1.4.2.14.21

Table 2.1-89: next_available_tbuffer Information.

2.1.4.2.14.14 bytes_to_trcds

Given the tape size in *bytes*, this routine returns the size in tbuffers.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
bytes	unsigned long	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
retval	int	the number of tbuffers in the specific tape size

Called By	
Function	**Where Described**
tape_open	Sec. 2.1.4.2.14.16

Table 2.1-90: bytes_to_trcds Information.

2.1.4 2.14.15 ini_tape

This void routine initializes the tape package.

Called By	
Function	**Where Described**
ini_fiolib	Sec. 2.1.4.2.2.13

Table 2.1-91: ini_tape Information.

2.1.4.2.14.16 tape_open

This routine opens a tape file for reading or writing.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
path	pointer to char	Standard
size	unsigned long	Standard
header	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
fhdr	pointer to register fio_header	Sec. 2.1.4.1.2 fioheader.h
frcd	pointer to fio_record	Sec. 2.1.4.2.1 fiofile.h
i	int	Standard
ufd	int	Standard
num_private_tbuffers	int	Standard
retval	int	Standard
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
space	lint_t	Sec. 2.1.4.2.14 fiotape.c
Return Values		
Return Value	**Type**	**Meaning**
retval	int	procedure failed due to problem initializing tbuffers
FIO_OPEN_UNIX_ERROR	int	procedure failed due toUnix OPEN error
FIO_ALLOC_UNIX_ERROR	int	procedure failed due to inability to allocate space for fiolib buffers
FIO_NOT_ASYNCH_ERROR	int	procedure failed: device does not support asynchronous I/O
FIO_OK	int	procedure was successful
Errors		
Error	**Reason for Error**	
FIO_OPEN_UNIX_ERROR	Unable to open specified file due to Unix error	
FIO_ALLOC_UNIX_ERROR	Unable to allocate space for fiolib buffers	
FIO_NOT_ASYNCH_ERROR	Device does not support asynchronous I/O	
FIO_WARNING_ERROR	- Only some tape buffers available - Unable to lockin file header tbuffer	

Calls	
Function	**Where Described**
ini_tbuffers	Sec. 2.1.4.2.14.12
IS_READABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
fio_alloc	Sec. 2.1.4.2.8.1
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
bytes_to_trcds	Sec. 2.1.4.2.14.14
new_tbq	Sec. 2.1.4.2.14.7
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
fio_error	Sec. 2.1.4.2.4.1
dequeue_tb	Sec. 2.1.4.2.14.9
enqueue_tb	Sec. 2.1.4.2.14.8
tape_close	Sec. 2.1.4.2.14.36
fio_free	Sec. 2.1.4.2.8.2
read_fiofile_header	Sec. 2.1.4.2.14.19
WAIT_FOR_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
promote_status_info	Sec. 2.1.4.2.14.5
read_stream	Sec. 2.1.4.2.14.21
FIO_RECORD_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape c (macro definition)
FIO_RECORD_PACKETS	Sec. 2.1.4.2.1 fiofile.h (macro definition)
COPY_FIOHDR	Sec. 2.1.4.1.2 fioheader.h (macro definition)
set_version_chksum_fiohdr	Sec. 2.1.4.2.3.20
write_fiofile_header	Sec. 2.1.4.2.14.25
VOLUME_NUMBER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
get_tape_number_fiohdr	Sec. 2.1.4.2.3.17

Table 2.1-92: tape_open Information.

2.1.4.2.14.17 tape_concat

This routine concatenates the tape file specified in *path* with the open file specified in *fdptr*, checking for consistancy. **tape_concat**() is called from the "fiolib.c" routine, **fio_concat**(). This routine provides the basis for the dual tape logger. By concatenating two tape drives and swapping tape reels, the user can create arbitrarily long data files. Each tape reel corresponds to a data file volume.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
path	pointer to char	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
ufd	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_OK	int	the procedure was successful
FIO_MAX_VOLUMES_ ERROR	int	procedure failed: maximum number of file volumes exceeded
FIO_OPEN_UNIX_ERROR	int	unable to open specified file due to Unix error

Errors	
Error	**Reason for Error**
FIO_MAX_VOLUMES_ERROR	Maximum number of file volumes exceeded
FIO_OPEN_UNIX_ERROR	Unable to open specified file due to Unix error

Calls	
Function	**Where Described**
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
IS_READABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)

Called By	
Function	**Where Described**
fio_concat	Sec. 2.1.4.2.2.6

Table 2.1-93: tape_concat Information.

2.1.4.2.14.18 tape_read

The routines for reading input fio_packets from fio_files are called from the "fiolib.c" routine **fio_read**(). **fio_read**() calls the tape routine **tape_read**() to read input data packets from the specified fio_file. The fio_packets are passed back to the user one at a time until the current tbuffer has been exhausted. When there are no more fio_packets left in the current tbuffer, **read_tbuffer**() is called to get another full tbuffer from the linked list of tbuffers. If the linked list of filled tbuffers has fallen below the TAPE_READ_THRESHOLD, the **read_stream**() routine is called. **read_stream**() takes advantage of the tape controller's ability to do multiple asynchronous read operations and reads a stream of fio_packets from the current tape drive into the linked list of tbuffers. When the end of the tape volume is reached, **read_stream**() calls **read_controller**() to swap tape volumes.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
pktptr	pointer to pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
copy	BOOL	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cvt_buffer [MAX_FIO_PACKET_SIZE]	char	Standard
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
frcd	pointer to register fio_record	Sec. 2.1.4.2.1 fiofile.h
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
nbytes	int	Standard
stat	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
stat	int	if stat = FIO_OK, the procedure was successful, if stat = an error message, the procedure failed
FIO_NO_PACKETS_AVAIL_ ERROR	int	no fio_packets available for reading

Errors	
Error	**Reason for Error**
FIO_EOF_ERROR	End of File
FIO_DRIVE_NOT_READY_ERROR	Tape drive for continuous concatenated file is not ready
FIO_NO_PACKETS_AVAIL_ERROR	No fio_packets available for reading
FIO_INTERNAL_ERROR	Corrupt fio_record

Calls	
Function	**Where Described**
enqueue_tb	Sec. 2.1.4.2.14.8
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
read_tbuffer	Sec. 2.1.4.2.14.20
IS_NBLOCKING	Sec. 2.1.4.1.1 fiolib.h (macro definition)
WAIT_FOR_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
IS_VERSION1	Sec. 2.1.4.1.1 fiolib.h (macro definition)
FIO_RECORD_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.2.1 fiofile.h (macro definition)
fio_error	Sec. 2.1.4.2.4.1
FIO_RECORD_PACKETS	Sec. 2.1.4.2.1 fiofile.h (macro definition)
cvt_fio_packet	Sec. 2.1.4.2.6.1
IS_VERSION	Sec. 2.1.4.1.1 fiolib.h (macro definition)
FIO_PACKET_PTR	Sec. 2.1.4.1.1 fiolib.h (macro definition)
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)

Called By	
Function	**Where Described**
fio_read	Sec. 2.1.4.2.2.7

Table 2.1-94: tape_concat Information.

2.1.4.2.14.19 read_fiofile_header

This routine makes an asynchronous IOCTL call to read in the first tape record, which contains the DataProbe record and the user defined header.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
ufd	int	Standard
ast	int	Standard
Errors		
Error	**Reason for Error**	
FIO_INTERNAL_ERROR	Read fio_file header error	
Calls		
Function	**Where Described**	
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
fio_error	Sec. 2.1.4.2.4.1	
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
Called By		
Function	**Where Described**	
read_controller	Sec. 2.1.4.2.14.22	
tape_open	Sec. 2.1.4.2.14.16	
tape_seek	Sec. 2.1.4.2.14.30	

Table 2.1-95: read_fiofile_header Information.

2.1.4.2.14.20 read_tbuffer

This routine attempts to get a full tbuffer of fio_packets. If successful, it returns a pointer to the tbuffer, if it fails it returns NULL. If there are less than TAPE_READ_THRESHOLD tbuffers, this routine calls **read_stream()** to get more. **read_tbuffer()** is called from **tape_read()** when no more fio_packets are in the current tbuffer. Note that this routine may have the effect of causing the high level status of the tape file to change.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
ast	int	Standard

Return Values		
Return Value	**Type**	**Meaning**
tbuf	pointer to tbuffer_t	if *tbuf* equals NULL, the routine failed; otherwise, *tbuf* points to the full tbuffer of fio_packets read

Errors	
Error	**Reason for Error**
FIO_EOV_ERROR	End of file volume
FIO_INTERNAL_ERROR	ll_status
FIO_EOF_ERROR	End of file
FIO_DRIVE_NOT_READY_ERROR	Tape drive for continous concatenated file is not ready

Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
dequeue_tb	Sec. 2.1.4.2.14.9
fio_error	Sec. 2.1.4.2.4.1
do_eotv_action	Sec. 2.1.4.2.14.2
read_stream	Sec. 2.1.4.2.14.21
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)

Called By	
Function	**Where Described**
tape_concat	Sec. 2.1.4.2.14.17
tape_seek	Sec. 2.1.4.2.14.30

Table 2.1-96: read_tbuffer Information.

2.1.4.2.14.21 read_stream

This routine reads a stream of tbuffers from the current tape drive. It makes the IOCTL calls for an asynchronous tape read. Depending upon the fio_file state, **read_stream**() makes a variable number of IOCTL FIOAREAD calls. **read_stream**() is called when the linked list of filled tbuffers has fallen below the TAPE_READ_THRESHOLD.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
i	int	Standard
num_ops	int	Standard
ufd	int	Standard
ast	int	Standard

Errors	
Error	**Reason for Error**
FIO_INTERNAL_ERROR	

Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
MIN	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TUBFS_AVAILABLE	Sec. 2.1.4.2.14 fiotape.c (macro definition)
next_available_tbuffer	Sec. 2.1.4.2.14.13
fio_error	Sec. 2.1.4.2.4.1
is_next_tape_volume	Sec. 2.1.4.2.14.6
read_controller	Sec. 2.1.4.2.14.22
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)

Called By	
Function	**Where Described**
read_tbuffer	Sec. 2.1.4.2.14.20
tape_seek	Sec. 2.1.4.2.14.30
tape_open	Sec. 2.1.4.2.14.16

Table 2.1-97: read_stream Information.

2.1.4.2.14.22 read_controller

This routine is called when **read_stream**() reaches the End of Tape Volume. **read_controller**() attempts to switch tapes. Once the new tape volume is mounted, this routine clears the drive status, allowing **read_stream**() to continue.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
status	struct mtget	Sec. 2.1.4.2.14 fiotape.c
vol	int	Standard
ufd	int	Standard
phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
ast	int	Standard

Errors	
Error	**Reason for Error**
RAW_DRIVE_NOT_READY_ERROR	
FIO_DRIVE_NOT_READY_ERROR	Tape drive for continuous concatenated file is not ready
FIO_INTERNAL_ERROR	No next tape volume
Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
is_next_tape_volume	Sec. 2.1.4.2.14.6
XYLOGICS_472_DRIVE_READY	Sec. 2.1.4.2.14 fiotape.c (macro definition)
init_eotv_action	Sec. 2.1.4.2.14.1
read_fiofile_header	Sec. 2.1.4.2.14.19
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
eotv_phase_action	Sec. 2.1.4.2.14.3
fio_error	Sec. 2.1.4.2.4.1
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
Called By	
Function	**Where Described**
read_stream	Sec. 2.1.4.2.14.21

Table 2.1-98: read_controller Information.

2.1.4.2.14.23 read_ast_handler

This routine is the high level AST handler for FIOAREADs. The AST handler determines if the read operation was successful. If it was, the tbuffer is freed to be used again. If the operation was not successful, error status codes are set.

The argument *pc* is the pc at the start of this function. The argument *prio* is the priority of the AST that called.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
pc	int	Standard
prio	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
errno	exter int	Standard
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
asynch	pointer to register struct_io	Standard
is_eotv	BOOL	Standard
fhdr	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h
frcd	pointer to fio_record	Sec. 2.1.4.2.1 fiofile.h
i	int	Standard
ast	int	Standard

Errors	
Error	**Reason for Error**
RAW_ABNORMAL_EOV_ERROR	
RAW_ABNORMAL_EOF_ERROR	
FIO_INTERNAL_ERROR	- Corrupt fio_record ignored - Volume is out of sequence - Unexpected End of Tape encountered while reading tape volume
RAW_EOV_ERROR	
FIO_VERSION1_ERROR	This fio file volume has an invalid version ID (Version1)
FIO_VERSION2_ERROR	This fio file volume has an invalid version ID (Version2)

Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
IS_VERSION1	Sec. 2.1.4.1.1 fiolib.h (macro definition)
FIO_RECORD_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.2.1 fiofile.h (macro definition)
fio_error	Sec. 2.1.4.2.4.1
enqueue_tb	Sec. 2.1.4.2.14.8
FIO_HEADER_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
get_version_chksum_fiohdr	Sec. 2.1.4.2.3.21
IS_VERSION2	Sec. 2.1.4.1.1 fiolib.h (macro definition)
VOLUME_NUMBER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
get_tape_number_fiohdr	Sec. 2.1.4.2.14.17
SET_VOLUME_STARTTIME	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_RECORD_STARTTIME	Sec. 2.1.4.2.14 fiotape.c (macro definition)
is_next_tape_volume	Sec. 2.1.4.2.14.6
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)

Table 2.1-99: read_ast_handler Information.

2.1.4.2.14.24 tape_write

The routines for writing input fio_packets are called from the "fiolib.c" routine **fio_write()**. **fio_write()** calls the tape routine **tape_write()** to write fio_packets. The fio_packets are stored in a tape buffer (tbuffer). When the current tbuffer is filled, **write_tbuffer()** is called to add the full tbuffer into the linked list of full tbuffers. When the TAPE_WRITE_THRESHOLD of tbuffers has been reached, the **write_stream()** routine is called. **write_stream()** takes advantage of the tape controller's ability to do multiple asynchronous write operations and writes a stream of tbuffers from the linked list of tbuffers to the tape drive. If the end of the tape volume is reached during **write_stream()**, the **write_controller()** routine is called to swap tapes.

Any buffers which get filled while the tape is streaming can be written immediately, however, once the tape stops streaming, nothing is written until the TAPE_WRITE_TRESHOLD of tbuffers is reached. In addition, there are complications at tape boundaries (i.e. End of Tape error recovery), especially in the case of continuous tape files.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
frcd	pointer to fio_record	Sec. 2.1.4.2.1 fiofile.h
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
nbytes	int	Standard
psize	int	Standard
stat	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
stat	int	*stat* is equal to the high level status; if equal to FIO_OK, then procedure was successful; if equal to FIO_EOF_ERROR, then procedure failed due to end of file error.
FIO_NO_BUFFERS_ERROR	int	procedure failed; no buffers available for writing
Errors		
Error	**Reason for Error**	
FIO_EOF_ERROR	End of file	
FIO_DRIVE_NOT_READY	Tape drive for continuous contcatenated file is not ready	
FIO_NO_BUFFERS_ERROR	No buffers available for writing	
Calls		
Function	**Where Described**	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
FIO_RECORD_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.2.1 fiofile.h (macro definition)	
write_tbuffer	Sec. 2.1.4.2.14.26	
next_available_tbuffer	Sec. 2.1.4.2.14.13	
FIO_RECORD_PACKETS	Sec. 2.1.4.2.1 fiofile.h (macro definition)	
IS_NBLOCKING	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
WAIT_FOR_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
write_stream	Sec. 2.1.4.2.14.27	
Called By		
Function	**Where Described**	
fio_write	Sec. 2.1.4.2.2.8	

Table 2.1-100: tape_write Information.

2.1.4.2.14.25 write_fiofile_header

This routine makes an asynchronous IOCTL call to write the DataProbe record and the user defined header. The fio_file header should always be the first tbuffer on a tape reel.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ufd	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
ufd	int	Standard
ast	int	Standard
Errors		
Error	**Reason for Error**	
FIO_INTERNAL_ERROR	Write fio_file header error	
Calls		
Function	**Where Described**	
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
fio_error	Sec. 2.1.4.2.4.1	
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
Called By		
Function	**Where Described**	
write_controller	Sec. 2.1.4.2.14.28	
tape_open	Sec. 2.1.4.2.14.16	

Table 2.1-101: write_fiofile_header Information.

2.1.4.2.14.26 write_tbuffer

This routine puts a full tbuffer of fio_packets into the linked list of tbuffers. **write_tbuffer()** is called from **tape_write()** when the current tbuffer is filled. Note that this routine may have the effect of causing the high level status of the tape file to change.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
frcd	pointer to fio_record	Sec. 2.1.4.2.1 fiofile.h
ufd	int	Standard
ast	int	Standard

Errors	
Error	**Reason for Error**
FIO_EOV_ERROR	End of fio file volume
RAW_EOV_ERROR	
FIO_EOF_ERROR	End of fio file
RAW_EOF_ERROR	
Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_RECORD_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
SET_VOLUME_STARTTIME	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_RECORD_STARTTIME	Sec. 2.1.4.2.14 fiotape.c (macro definition)
enqueue_tb	Sec. 2.1.4.2.14.8
do_eotv_action	Sec. 2.1.4.2.14.2
is_next_tape_volume	Sec. 2.1.4.2.14.6
write_stream	Sec. 2.1.4.2.14.27
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
Called By	
Function	**Where Described**
tape_write	Sec. 2.1.4.2.14.24
tape_close	Sec. 2.1.4.2.14.36

Table 2.1-102: write_tbuffer Information.

2.1.4.2.14.27 write_stream

This routine writes a stream of tbuffers from the current tape drive. **write_stream**() makes a variable number of IOCTL FIOAWRITE calls, depending upon the state of the tape file (e.g. are there any errors, is the tape drive streaming, how many buffers are filled, and how many write operations are still pending from the physical tape controller).

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to register tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
num_ops	int	Standard
max_ops	int	Standard
i	int	Standard
ufd	int	Standard
ast	int	Standard

Errors	
Error	**Reason for Error**
RAW_EOF_ERROR	
RAW_ABNORMAL_ERROR	
RAW_EOV_ERROR	
RAW_ABNORMAL_EOV_ERROR	
RAW_DRIVE_NOT_READY_ERROR	
FIO_INTERNAL_ERROR	Write fio_file header error
Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
MIN	Sec. 2.1.4.2.14 fiotape.c (macro definition)
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
write_controller	Sec. 2.1.4.2.14.28
dequeue_tb	Sec. 2.1.4.2.14.9
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
fio_error	Sec. 2.1.4.2.4.1
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
Called By	
Function	**Where Described**
tape_write	Sec. 2.1.4.2.14.24
write_tbuffer	Sec. 2.1.4.2.14.26
tape_close	Sec. 2.1.4.2.14.36
write_controller	Sec. 2.1.4.2.14.28

Table 2.1-103: write_stream Information.

2.1.4.2.14.28 write_controller

This routine is called to switch tapes when **write_stream()** reaches the End of Tape Volume. Note that the tapes cannot be switched until all IOCTL calls are returned.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
status	struct mtget	Standard
tbq	pointer to tbuffer_q	Sec. 2.1.4.2.14 fiotape.c
ffhdr	pointer to fio_header	Sec. 2.1.4.1.3 fioheader.h
phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
vol	int	Standard
ufd	int	Standard
ast	int	Standard

Errors	
Error	**Reason for Error**
RAW_ABNORMAL_EOV_ERROR	
RAW_DRIVE_NOT_READY_ERROR	
RAW_ABNORMAL_EOF_ERROR	
FIO_DRIVE_NOT_READY_ERROR	Tape drive for continuous concatenated file is not ready
FIO_INTERNAL_ERROR	No next tape volume
FIO_ALLOC_UNIX_ERROR	Allocation error; lost packets
Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
is_next_tape_volume	Sec. 2.1.4.2.14.6
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
XYLOGICS_472_DRIVE_READY	Sec. 2.1.4.2.14 fiotape.c (macro definition)
init_eotv_action	Sec. 2.1.4.2.14.1
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
concat_tbqs	Sec. 2.1.4.2.14.10
fio_free	Sec. 2.1.2.2.8.2
null_tbq	Sec. 2.1.4.2.14.11
fio_error	Sec. 2.1.4.2.4.1
VOLUME_NUMBER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
set_tape_number_fiohdr	Sec. 2.1.4.2.3.16
write_fiofile_header	Sec. 2.1.4.2.14.25
write_stream	Sec. 2.1.4.2.14.27
eotv_phase_action	Sec. 2.1.4.2.14.3
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
Called By	
Function	**Where Described**
write_stream	Sec. 2.1.4.2.14.27

Table 2.1-104: write_controller Information.

2.1.4.2.14.29 write_ast_handler

This routine is the high level AST handler for FIOAWRITEs. The AST handler determines if the write operation was successful. If it was, the tbuffer is freed to be used again. If the operation was not successful, error status codes are set.

The argument *pc* is the pc at the start of this function. The argument *prio* is the priority of the AST that called.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
pc	int	Standard
prio	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
errno	extern int	Standard
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
asynch	pointer to register struct _io	Standard
ufd	int	Standard
ast	int	Standard

Errors	
Error	**Reason for Error**
FIO_INTERNAL_ERROR	- Unexpected End of Tape encountered while WRITING tape volume; or a write error due to bad tape - Unexpected Result; write only partly succeeded, but there was no error message.

Calls	
Function	**Where Described**
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
enqueue_tb	Sec. 2.1.4.2.14.8
fio_error	Sec. 2.1.4.2.4.1
is_next_tape_volume	Sec. 2.1.4.2.14.6
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)

Table 2.1-105: write_ast_handler Information.

2.1.4.2.14.30 tape_seek

This routine seeks to a specified location in a tape file. The seeking is accomplished in terms of network time ticks. First, the time offset is translated into an absolute exercise time. Then, a check is made to see that all IOCTL calls are returned. An attempt is made to find the volume on which the seek time is most likely to be found. A check is made to see if the volume is actually mounted. Then, streaming is started. A search is made for the correct buffer, being careful about reaching the End of File. Then, the routine backs up to an appropriate point to start the seek forward (avoiding backing up past the fio_header) and checks for the correct tbuffer.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
offset	int	Standard
whence	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
frcd	pointer to fio_record	Sec. 2.1.4.2.1 fiofile.h
i	int	Standard
tgtvol	int	Standard
tgttime	int	Standard
curtime	int	Standard
starttime	int	Standard
tgt_ufd	int	Standard
cur_ufd	int	Standard
tgt_status	struct mtget	Standard
cur_status	struct mtget	Standard
retval	int	Standard
hbound	int	Standard
lasttime	int	Standard
total_backed_up	int	Standard
trcds_to_start	int	Standard
bot	BOOL	Standard
volume	int	Standard
phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
action	eotv_e	Sec. 2.1.4.2.14 fiotape.c

Return Values		
Return Value	**Type**	**Meaning**
FIO_SEEK_RELATIVE_ TAPE_END_ERROR	int	the fio_seek_end *whence* value is invalid for tape files
FIO_SEEK_INVALID_ WHENCE_VALUE_ERROR	int	seek called with an invalid value for the *whence* parameter
FIO_SEEK_PAST_EOF_ ERROR	int	attempt to seek past the end of the specified file volume
retval	int	if retval = FIO_OK, the procedure was successful; if retval = an error message, the procedure failed

Errors	
Error	**Reason for Error**
FIO_SEEK_PAST_BOF_ERROR	Attempt to seek past the beginning of the specified file volume
FIO_SEEK_RELATIVE_TAPE_END_ERROR	The fio_seek_end *whence* value is invalid for tape files
FIO_SEEK_INVALID_WHENCE_VALUE_ ERROR	Seek called with an invalid value for the *whence* parameter
FIO_SEEK_VOLUME_NOT_AVAILABLE_ ERROR	Cannot seek to the specified exercise time. Volume is not available
FIO_INTERNAL_ERROR	- Invalid eotv_phase - eotv_volume not tgtvol - tgttime<curtime and tgtvol>curvol - eotv_volume is tgt volume - ll_status - Starting the search after the target time - filled_tbq is unexpectedly empty
FIO_DRIVE_NOT_READY_ERROR	Tape drive for continuous concatenated tape files is not ready
FIO_SEEK_PAST_EOF_ERROR	Attempt to seek past the end of the specified file volume

Calls	
Function	**Where Described**
TAPEINFO_PTRR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
VOLUME_STARTTIME	Sec. 2.1.4.2.14 fiotape.c (macro definition)
WAIT_FOR_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
tvolume_starttime	Sec. 2.1.4.2.14.33
fio_error	Sec. 2.1.4.2.4.1
IS_CONTINUOUS	Sec. 2.1.4.1.1 fiolib.h (macro definition)
tvolume_available	Sec. 2.1.4.2.14.32
VOLUME_NUMBER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
enqueue_tb	Sec. 2.1.4.2.14.8
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
dequeue_tb	Sec. 2.1.4.2.14.9
FIO_RECORD_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)
FIO_RECORD_PACKETS	Sec. 2.1.4.2.1 fiofile.h (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
XYLOGICS_472_DRIVE_READY	Sec. 2.1.4.2.14 fiotape.c (macro definition)
eotv_phase_action	Sec. 2.1.4.2.14.3
XYLOGICS_472_DRIVE_ONLINE	Sec. 2.1.4.2.14 fiotape.c (macro definition)
backup_past_target	Sec. 2.1.4.2.14.34
read_fiofile_header	Sec. 2.1.4.2.14.19
read_stream	Sec. 2.1.4.2.14.21
WAIT_FOR_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
read_tbuffer	Sec. 2.1.4.2.14.20
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.2.1 fiofile.h (macro definition)
FIO_RECORD_STARTTIME	Sec. 2.1.4.2.14 fiotape.c (macro definition)
QUEUE_HEAD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)

Called By	
Function	**Where Described**
fio_seek	Sec. 2.1.4.2.2.9

Table 2.1-106: tape_seek Information.

2.1.4.2.14.31 tape_seek_time

This routine returns the time of the current packet.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c
Return Values		
Return Value	**Type**	**Meaning**
FIO_PACKET_TIME (tptr->current_packet)	long	the time of the current packet
0	int	procedure failed
Calls		
Function	**Where Described**	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
Called By		
Function	**Where Described**	
fio_seek	Sec. 2.1.4.2.2.9	

Table 2.1-107: tape_seek_time Information.

2.1.4.2.14.32 tvolume_available

This routine returns TRUE when the desired volume is mounted. Note that this procedure should only be called for CONTINUOUS files.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
vol	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fhdr	fio_header	Sec. 2.1.4.1.3 fioheader.h
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c
status	struct mtget	Standard
ufd	int	Standard
volnum	int	Standard
phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
retval	BOOL	Standard

Return Values		
Return Value	**Type**	**Meaning**
retval	BOOL	retval = TRUE if the volume specified in *vol* is mounted; retval = FALSE if the volume is not mounted

Errors	
Error	**Reason for Error**
FIO_INTERNAL_ERROR	Read of fio_header failed

Calls	
Function	**Where Described**
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
XYLOGICS_472_DRIVE_READY	Sec. 2.1.4.2.14 fiotape.c (macro definition)
fioerror	Sec. 2.1.4.2.4.1
get_version_chksum_fiohdr	Sec. 2.1.4.2.3.21
get_tape_number_fiohdr	Sec. 2.1.4.2.3.17
VOLUME_NUMBER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
eotv_phase_action	Sec. 2.1.4.2.14.3
XYLOGICS_472_DRIVE_ONLINE	Sec. 2.1.4.2.14 fiotape.c (macro definition)

Called By	
Function	**Where Described**
tape_seek	Sec. 2.1.4.2.14.30

Table 2.1-108: tvolume_available Information.

2.1.4.2.14.33 tvolume_starttime

This routine sets a pointer to the *starttime* of the volume specified in *vol.*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
vol	int	Standard
starttime	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
fhdr	fio_header	Sec. 2.1.4.2.1 fiofile.h
frcd	fio_record	Sec. 2.1.4.2.1 fiofile.h
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c
ufd	int	Standard
volnum	int	Standard
retval	int	Standard
phase	eotv_phs_e	Sec. 2.1.4.2.14 fiotape.c
status	struct mtget	Standard

Return Values		
Return Value	**Type**	**Meaning**
FIO_SEEK_VOLUME_NOT_AVAILABLE_ERROR	int	The procedure failed
retval	int	If retval = FIO_OK, the procedure was successful; If retval = an error message, the procedure failed

Errors	
Error	**Reason for Error**
FIO_INTERNAL_ERROR	- Read of *fhdr* failed - Read of *frcd* failed
FIO_SEEK_VOLUME_NOT_AVAILABLE_ERROR	Cannot seek to the specified exercise time; volume is not available
FIO_DRIVE_NOT_READY	Tape drive for continuous concatenated file is not ready

Calls	
Function	**Where Described**
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
XYLOGICS_472_DRIVE_READY	Sec. 2.1.4.2.14 fiotape.c (macro definition)
fio_error	Sec. 2.1.4.2.4.1
get_version_chksum_error	Sec. 2.1.4.2.3.21
get_tape_number_fiohdr	Sec. 2.1.4.2.3.17
eotv_phase_action	Sec. 2.1.4.2.14.1

Called By	
Function	**Where Described**
tape_seek	Sec. 2.1.4.2.14.30

Table 2.1-109: tvolume_starttime Information.

2.1.4.2.14.34 backup_past_target

This backs up the specified drive past the specified time, sets the fio_file pointer and returns the total number of records backed up.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ufd	int	Standard
starttime	int	Standard
tgttime	int	Standard
lasttime	int	Standard
trcds_to_start	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ms_to_start	int	Standard
ms_to_tgt	int	Standard
trcds_backed_up	int	Standard
frcd	pointer to fio_record	Sec. 2.1.4.2.1 fiofile.h

Return Values		
Return Value	**Type**	**Meaning**
trcds_backed_up	int	The total number of fio_records backed up

Errors	
Error	**Reason for Error**
FIO_INTERNAL_ERROR	Backup_to target: negative backup Backup_past_target: large MTBSR failed Backup_past_target: small MTBSR failed Backup_to_target read failed

Calls	
Function	**Where Described**
fio_error	Sec. 2.1.4.2.4.1
unix_tape_action	Sec. 2.1.4.2.14.4
FIO_RECORD_STARTTIME	Sec. 2.1.4.2.14 fiotape.c (macro definition)

Called By	
Function	**Where Described**
tape_seek	Sec. 2.1.4.2.14.30

Table 2.1-110: backup_past_target Information.

2.1.4.2.14.35 tape_getrawfd

This routine returns the raw Unix file descriptor for the volume specified in *volume_offset*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fdptr	fio_file_p	Sec. 2.1.4.2.1 fiofile.h
volume_offset	int	Standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to tape_t	Sec. 2.1.4.2.14 fiotape.c
vol	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
V OLUME_UFD(tptr, vol))	int	the raw Unix file descriptor
Calls		
Function	**Where Described**	
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)	
Called By		
Function	**Where Described**	
fio_getrawfd	Sec. 2.1.4.2.2.10	

Table 2.1-111: tape_getrawfd Information.

2.1.4.2.14.36 tape_close

This routine closes an open tape file. The state of the file is checked. If the file is opened for writing, all IOCTL calls must have been executed. The remaining filled_tbq tbuffers, the remaining private_tbq tbuffers, and the current_tbuffer are freed and memory allocations are cleaned up. This routine is called from the "fiolib.c" routine **fio_close**().

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ufd	int	Standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tptr	pointer to register tape_t	Sec. 2.1.4.2.14 fiotape.c
tbuf	pointer to tbuffer_t	Sec. 2.1.4.2.14 fiotape.c
frcd	pointer to fio_record	Sec. 2.1.4.2.1 fiofile.h
write_rcds	BOOL	Standard
ufd	int	Standard
i	int	Standard
retval	int	Standard
ast	int	Standard
Return Values		
Return Value	**Type**	**Meaning**
retval	int	If retval = FIO_OK, the procedure was successful; If retval = an error message, the procedure failed

Errors	
Error	**Reason for Error**
FIO_CLOSE_UNIX_ERROR	Error closing file
FIO_INTERNAL_ERROR	Tape_close: filled_tbq is not empty
Calls	
Function	**Where Described**
TAPEINFO_PTR	Sec. 2.1.4.2.14 fiotape.c (macro definition)
DISABLE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
IS_WRITABLE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
FIO_RECORD_OF_TBUFFER	Sec. 2.1.4.2.14 fiotape.c (macro definition)
FIO_RECORD_PACKETS	Sec. 2.1.4.2.1 fiofile.h (macro definition)
FIO_RECORD_BYTE_COUNT	Sec. 2.1.4.2.1 fiofile.h (macro definition)
write_tbuffer	Sec. 2.1.4.2.14.26
enqueue_tb	Sec. 2.1.4.2.14.8
QUEUE_COUNT	Sec. 2.1.4.2.14 fiotape.c (macro definition)
write_stream	Sec. 2.1.4.2.14.27
WAIT_FOR_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
RESTORE_AST	Sec. 2.1.4.2.14 fiotape.c (macro definition)
VOLUME_UFD	Sec. 2.1.4.2.14 fiotape.c (macro definition)
dequeue_tb	Sec. 2.1.4.2.14.9
fio_free	Sec. 2.1.4.2.5.2
Called By	
Function	**Where Described**
tape_open	Sec. 2.1.4.2.14.16
fio_close	Sec. 2.1.4.2.2.12

Table 2.1-112: tape_close Information.

2.2 Data Logger Files

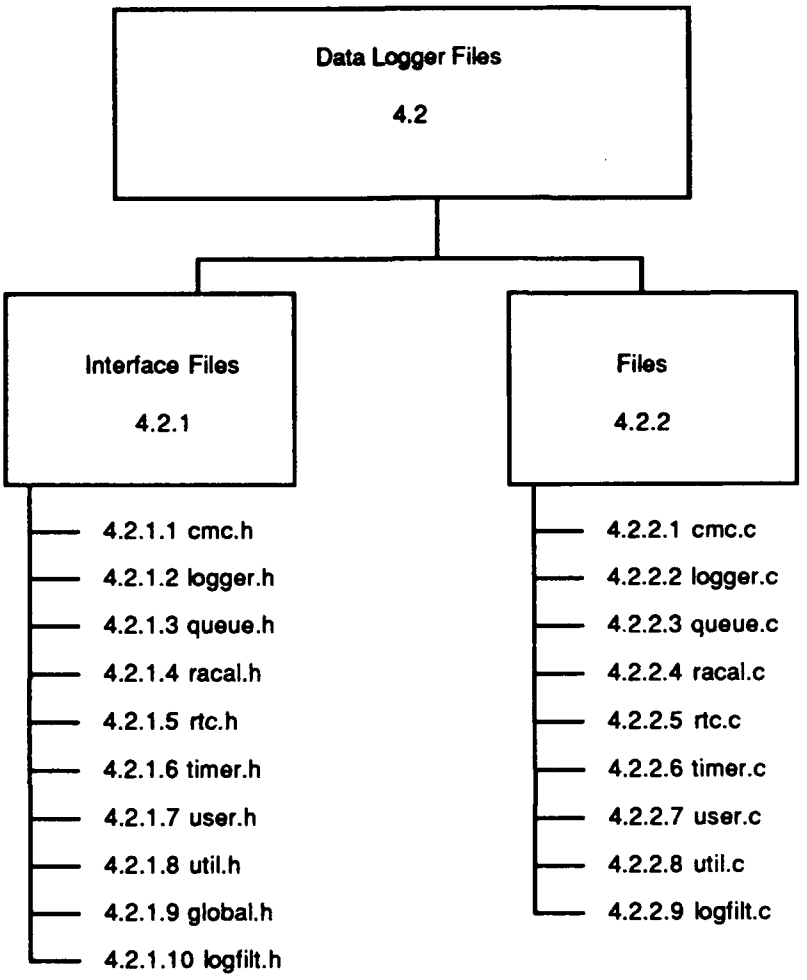


Figure 2-2: Data Logger Files Software Structure

2.2.1 Data Logger Header Files

2.2.1.1 cmc.h

cmc.h defines some cmc network specific routines.

Defines:

CMC_PATH
ACCEPT_ALL
REG_SHFT
REG_MASK
BPB
NUM_EXERCISE_IDS
NUM_EXERCISE_REGISTERS
ACCEPT_EXERCISE
NUM_PROTOCOL_REGISTERS

NUM_PROTOCOL_IDS
ACCEPT_PROTOCOL
proprietaryProtocolNumbers

Accounting Macros:
INC_SENDPVD_COUNT()
INC_RCVDPVD_COUNT()
INC_SEND_COUNT()
INC_RCVD_COUNT()
INC_LATE_COUNT()
INC_LATE_MARGIN()
INC_DROP_COUNT()
INC_FILTER_COUNT()

Global Variable Declarations:
sendpvd_count
rcvdpvd_count
send_count
rcvd_count
late_count
late_margin
drop_count
filter_count

External procedure declarations:
cmc_init()
cmc_term()
cmc_accept()
cmc_protocol_types()
cmc_zero_status()
cmc_report_status()
cmc_gettime16()
cmc_gettime32()
cmc_settime()

2.2.1.2 logger.h

This file defines the interface to the logger finite state machine. This file exports a number of global variables which act as flags for the finite state machine.

Includes "p_logger.h"

External procedure declarations:
logger_op_error()
logger_connect()
logger_disconnect()
logger_seek_relative()
logger_speed()
logger_stop()
logger_start()
logger_suspend()
logger_status()
logger_get_state()
logger_get_activity()

2.2.1.3 queue.h

"queue.h" is a low level module which provides a general purpose queueing facility.

Defines:
FREE_QUEUE
IN_QUEUE
OUT_QUEUE

External procedure definitions:
init_queues()
reinit_queues()
enqueue()
dequeue()
queue_count()

2.2.1.4 racal.h

The racal.h file defines the following:
CMC_MILLI_THRESH
WAIT_INIT_CHAR
GET_N_CHARS
GET_TERM_CHAR_1
GET_TERM_CHAR_2

External procedure declarations:
racal_close()
racal_open()
reset_racal_time()
racal_start_recording()
racal_start_playing()
racal_stop()
racal_seek_to_time()

2.2.1.5 rtc.h

This file handles the Run-Time Commands. When the logger is in interactive mode, run-time commands will be coming from STDIN. When the logger is in Server mode, run-time commands will be coming from the client over the CMC network.

Includes:
"p_assoc.h"
"p_logger.h"
"network.h"
"fiolib.h"

External Procedure Declarations:
init_rtc()
free_rtc()
logger_message_location()
logger_msg()
logger_error()
protocol_of()
rtc_size()

handle_rtc()
handle_remote_rtc()
new_rtc()
send_clock_tick()

2.2.1.6 timer.h

This file declares the external interface to "timer.c".

on_time()
set_fast_time_factor()
get_fast_time_factor()
set_time()
get_time()
freeze_time()
unfreeze_time()
time_string()

2.2.1.7 user.h

This file declares the interface to the User Interface file, "user.c".

External procedure declarations:

collect_racal_time_info()
collect_fiofile_info()
collect_cmcsubscribe_info()
collect_fioheader_info()
collect_playback_info()
continue_or_quit()
yes_or_no()

2.2.1.8 util.h

Defines:

MAX_VOLUMES
DISK_FILE
NET_FILE
TAPE_FILE

Typedefs:

FIOLE_INFO file_info

External Procedure Declaration:

init_file_info()

2.2.1.9 global.h

This file defines some global macros:

MIN
MAX
ABS

2.2.1.10 logfilt.h

This file defines **logger_filter()**, an external procedure used by the CMC logger filter routine and the exercise filtering mechanism of the logger.

2.2.2 Data Logger Files

2.2.2.1 cmc.c

This file provides an interface to services provided by the SIMNET network interface library (libnetif). The services used include the timer, statistics about the number of packets handled, and the downloadable filter, a mechanism for filtering out PDU's.

Includes:

<stdio.h>
<fcntl.h>
<sys/types.h>
"sim_defines.h"
"network.h"
"netfilter.h"
"nettab.h"
"enpioctl.h"
"global.h"
"rtc.h"
"cmc.h"

Standard Procedure Declarations:

free()
malloc()
perror()
time()

External Procedure Declaration:

cmc_filter_init()

Exported Global Variables:

sendpvd_count	--the number of RTC packets sent on the logger's client
rcvdpvd_count	--the number of RTC packets received from the logger's client
send_count	--the number of packets sent on the CMC network
rcvd_count	--the number of packets received from the CMC network
late_count	--the number of packets sent late
late_margin	--the margin by which packets are sent late
drop_count	--the number of packets dropped
filter_count	--the number of packets filtered
cmc_base_time	--used in conjunction with the 16-bit clock, the base time from which the 16-bit network clock is an offset.
cmcfd	--the UNIX file descriptor for dealing with CMC network.
msgbuf[256]	--the buffer for messages.

The next few routines, (2.2.2.1.1 through 2.2.2.1.4) constitute the interface to the downloadable filter. This filter (see logfilt.c) is loaded onto the ethernet card by a separate program (netstart). These routines provide a mechanism for the logger to communicate

with the filter during runtime. The filter is used to weed out unwanted packets at the board level when the logger is recording. The filter is written to filter out packets based on ExerciseID and on Protocol.

Defines:
ACCEPT_ALL_IDS
CMC_FILTER

2.2.2.1.1 cmc_filter_init

This routine initializes the filter on the CMC card. The filter is initialized to accept only logger RTC packets.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	standard
Calls		
Function	**Where Described**	
net_flush	Sec. 2.20.1 See MCC CSCI SDD	
net_reg_write	Sec. 2.20.1 See MCC CSCI SDD	
cmc_accept	Sec. 2.2.2.1.3	
Called By		
Function	**Where Described**	
cmc_init	Sec. 2.2.2.1.5	

Table 2.2-1: cmc_filter_init Information.

2.2.2.1.2 cmc_protocol_types

This routine initializes the list of protocol types which the logger will accept. Currently the logger will accept only simulation type, or voice type, or both.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sim_protocol	BOOL	standard
voice_protocol	BOOL	standard
Calls		
Function	**Where Described**	
net_init_type	Sec. 2.20.1 See MCC CSCI SDD	
net_add_type	Sec. 2.20.1 See MCC CSCI SDD	
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	

Table 2.2-2: cmc_protocol_types Information.

2.2.2.1.3 cmc_accept

This routine adds an exercise/protocol ID to the list to be accepted by the logger's CMC filter. Passing a value of ACCEPT_ALL is an indication that all exercise/protocol ID's should be accepted. This routine returns a 1 on successful completion and 0 if the operation fails.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
id	int	standard
is_exercise	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
reg_index	register int	standard
last_id	int	standard
num_reg	int	standard
reg_value	int	standard
i	int	standard
Return Values		
Return Value	**Type**	**Meaning**
0	int	The procedure failed
1	int	The procedure was successful
Errors		
Error	**Reason for Error**	
logger_error	Invalid ID value	
Calls		
Function	**Where Described**	
logger_error	Sec. 2.2.2.5.3	
net_reg_write	Sec. 2.20.1 See MCC CSCI SDD	
net_reg_read	Sec. 2.20.1 See MCC CSCI SDD	
Called By		
Function	**Where Described**	
cmc_filter_init	Sec. 2.2.2.1.1	
logger_accept_pdus	Sec. 2.2.2.2.12	

Table 2.2-3: cmc_accept Information.

2.2.2.1.4 cmc_reject

This routine removes an exercise/protocol ID from the list to be accepted by the logger's CMC filter. This routine returns 1 on successful completion and 0 if the operation fails.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
id	int	standard
is_exercise	int	standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
reg_index	register int	standard
last_id	int	standard
reg_value	int	standard
Return Values		
Return Value	**Type**	**Meaning**
0	int	The procedure failed
1	int	The procedure was successful
Errors		
Error	**Reason for Error**	
logger_error	Invalid ID value	
Calls		
Function	**Where Described**	
logger_error	Sec. 2.2.2.5.3	
net_reg_write	Sec. 2.20.1 See MCC CSCI SDD	
net_reg_read	Sec. 2.20.1 See MCC CSCI SDD	
Called By		
Function	**Where Described**	
logger_accept_pdus	Sec. 2.2.2.2.12	

Table 2.2-4: cmc_reject Information.

2.2.2.1.5 cmc_init

This routine initializes the CMC network in the appropriate modes:

sendpvd_count = 0
rcvpvd_count = 0
send_count = 0
rcvd_count = 0
late_count = 0
drop_count = 0
filter_count = 0
cmc_base_time = 0

Parameters		
Parameter	**Type**	**Where Typedef Declared**
id	int	standard
is_exercise	int	standard
Return Values		
Return Value	**Type**	**Meaning**
0	int	The procedure failed
1	int	The procedure was successful

Calls	
Function	**Where Described**
cmc_filter_init	Sec. 2.2.2.1.1
logger_init_cmc	Sec. 2.2.2.2.11

Table 2.2-5: cmc_init Information.

2.2.2.1.6 cmc_term

This routine terminates the CMC network in the appropriate modes. The routine returns 1 on successful completion and 0 if the operation fails.

Return Values		
Return Value	**Type**	**Meaning**
0	int	The procedure failed
1	int	The procedure was successful

Table 2.2-6: cmc_term Information.

2.2.2.1.7 cmc_zero_status

This routine zeroes the statistics about the network which are being kept by the network library (libnetif). The routine returns 1 on successful completion and 0 if the operation fails.

Calls	
Function	**Where Described**
net_zero_statistics	Sec. 2.20.1 See MCC CSCI SDD
Called By	
Function	**Where Described**
logger_continue	Sec. 2.2.2.2 logger.c

Table 2.2-7: cmc_zero_status Information.

2.2.2.1.8 cmc_report_status

This routine reports the statistics about the network which the network library has been keeping.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
recording	BOOL	standard
play_time	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
status_buffer[N_STATS]	int	standard
seconds	float	standard
status_string[80]	char	standard
stat	int	standard

Calls	
Function	**Where Described**
net_get_characteristics	Sec. 2.20.1 See MCC CSCI SDD
logger_msg	Sec. 2.2.2.5.2
net_stat_string	Sec. 2.20.1 See MCC CSCI SDD
Called By	
Function	**Where Described**
ccatcher	Sec. 2.2.2.2.34
logger_stop	Sec. 2.2.2.2.28

Table 2.2-8: cmc_report_status Information.

The next few routines (2.2.2.1.9 through 2.2.2.1.11) are low level routines for getting and setting the clocks on the ethernet card.

2.2.2.1.9 cmc_gettime16

This routine gets the value of the 16 bit timer from the CMC card.
One count = 998.26 microseconds.

Return Values		
Return Value	**Type**	**Meaning**
net_current_time(cmfd) + cmc_base_time	int	The value of the 16 bit timer from the CMC card
Calls		
Function	**Where Described**	
net_current_time	Sec. 2.20.1 See MCC CSCI SDD	
Called By		
Function	**Where Described**	
on_time	Sec. 2.2.2.6.10	

Table 2.2-9: cmc_gettime16 Information.

2.2.2.1.10 cmc_gettime32

This routine gets the value of the 32 bit timer from the CMC card.
One count = 998.26 microseconds.

Return Values		
Return Value	**Type**	**Meaning**
net_gettime(cmfd)	int	The value of the 32 bit timer from the CMC card
Calls		
Function	**Where Described**	
net_gettime	Sec. 2.20.1 See MCC CSCI SDD	

Called By	
Function	**Where Described**
get_time	Sec. 2.2.2.6.7
unfreeze_time	Sec. 2.2.2.6.9
freeze_time	Sec. 2.2.2.6.8

Table 2.2-10: cmc_gettime32 Information.

2.2.2.1.11 cmc_settime

This routine sets the values of both the 16-bit and 32-bit timers from the CMC card. One count = 998.26 microseconds. The parameter *ntime* is the new value for the 32-bit timer on the CMC card.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
ntime	int	standard
Calls		
Function	**Where Described**	
net_init_time	Sec. 2.20.1 See MCC CSCI SDD	
net_settime	Sec. 2.20.1 See MCC CSCI SDD	
Called By		
Function	**Where Described**	
reset_time	Sec. 2.2.2.6.5	
set_time	Sec. 2.2.2.6.6	
set_fast_time_factor	Sec. 2.2.2.6.3	

Table 2.2-11: cmc_settime Information.

2.2.2.2 logger.c

This is the primary file for the Data Logger. The basic structure of the Data Logger involves an initialization phase during which the Data Logger is placed in either stand-alone or server mode and then initialized for recording, playback, or copying. The Data Logger then enters the main loop of the program. In the main loop, the Data Logger takes care of two primary functions: (1) to aquire and process packets, and (2) to respond to run-time commands.

Includes:

<signal.h>
>stdio.h>
<sys/ioctl.h>
<sys/types.h>
<sys/timeb.h>
<sys/times.h>
<time.h>

"fiolib.h"
"fioheader.h"
"fioerror.h"
"network.h"
"p_num.h"
"p_logger.h"
"global.h"
"cmc.h"
"logger.h"
"logtime.h"
"queue.h"
"racal.h"
"rtc.h"
"timer.h"
"user.h"
"util.h"

Standard Procedures:

perror()
exit()
strcat()
strncpy()

Static Procedures:

process_parms()
init_state_variables()
init_logger_variables()
logger_init()
init_logger_for_server()
init_logger_for_copying()
init_logger_for_playback()
init_logger_for_recording()
open_fio_files
logger_accept_pdus()
logger_init_cmc()
logger_get_remote_rtc()
logger_put_remote_rtc()
logger_fsm()
logger_copying_state()
logger_playback_state()
logger_recording_state()
logger_reset()
logger_clock_tick()
print_advice()
ccatcher()
bells()
print_command_line_help()

Defines:

Alarm Level:

NOTE
ALARM
PANIC

DataProbe Header:
- TAPE_TYPE
- RUN_NUM
- START_TAPE_NUM
- BPW
- HDR_SIZE
- DATE_OF_COPY

- CYCLES_PER_RTC_MASK
- LATE_MARGIN
- DRIVE_NOT_READY_CRISIS_BOUND
- MAX_FAILED_READS
- MAX_DNR_ERRORS
- CLOCKTICK_INTERVAL

Declarations:

Logger State Variables:
- logger_state
- logger_activity
- logger_operation
- logger_activity_started

Static Variables:
- control_c
- debug
- maxprio
- query_all
- racal

Global File Variables:
- rd_files
- wr_files
- fio_hdr
- readfd
- writefd
- cmcfd

Fio_header String Defines and Variables:
- FILEID_SZ
- file_id
- command_str
- dp_comment
- dp_project
- start_date_str
- start_time_str

Time on Racal clock (translated to CMC msec) when CMC clock reads 0 Variables:
- racal_time_at_start
- start_datetime
- current_timestamp
- copied_count

msgbuf
saved_packet
write_packet

Filtering on Playback Macros and Variables:
playback_stealth_protocol
InitPlaybackFilter()
FilterProtocol()

2.2.2.2.1 main

main() is the entry point of the Data Logger program.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
argc	int	standard
argv[]	pointer to char	standard
Return Values		
Return Value	**Type**	**Meaning**
0	int	The operation was successful
Calls		
Function	**Where Described**	
init_state_variables	Sec. 2.2.2.2.3	
init_queues	Sec. 2.2.2.3.4	
init_rtc	Sec. 2.2.2.5.6	
init_user	Sec. 2.2.2.7.13	
logger_message_location	Sec. 2.2.2.5.1	
process_parms	Sec. 2.2.2.2.2	
logger_msg	Sec. 2.2.2.5.2	
yes_or_no	Sec. 2.2.2.7.9	
init_logger_variables	Sec. 2.2.2.2.4	
logger_init	Sec. 2.2.2.2.5	
logger_fsm	Sec. 2.2.2.2.15	
logger_reset	Sec. 2.2.2.2.20	

Table 2.2-12: main Information.

2.2.2.2.2 process_parms

This routine processes the command-line flags.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
argc	int	standard
argv[]	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
progname	pointer to char	standard
i	int	standard

Errors	
Error	**Reason for Error**
logger_error	- Specifying activity not allowed in LoggerServer mode - Cannot copy and record simultaneously - Cannot copy and play simultaneously - Cannot play and record simultaneously - Cannot play and copy simultaneously - Cannot record and play simultaneously - Cannot record and copy simultaneously - Unknown switch - Logger must copy, play, or record

Calls	
Function	**Where Described**
logger_msg	Sec. 2.2.2.5.2
print_command_line_help	Sec. 2.2.2.2.36
logger_error	Sec. 2.2.2.5.3

Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1

Table 2.2-13: process_parms Information.

2.2.2.2.3 init_state_variables

This routine initializes the state variables as follows:

logger_activity:	loggerNoActivity
logger_operation:	loggerInteractive
logger_state:	LoggerDisconnected
logger_activity_started:	FALSE
control_c:	FALSE
debug:	FALSE
maxprio:	FALSE
query_all:	FALSE
racal:	FALSE

Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1

Table 2.2-14: init_state_variables.

2.2.2.2.4 init_logger_variables

This routine initializes the logger variables.

Calls	
Function	**Where Described**
init_file_info	Sec. 2.2.2.8.1
null_fiohdr	Sec. 2.1.4.2.3.23

Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1

Table 2.2-15: init_logger_variables Information.

2.2.2.2.5 logger_init

This routine calls different initialization routines depending on the logger activity: recording, playback, or copying.

Calls	
Function	**Where Described**
reinit_queues	Sec. 2.2.2.3.5
init_logger_for_server	Sec. 2.2.2.2.6
init_logger_for_copying	Sec. 2.2.2.2.7
init_logger_for_playback	Sec. 2.2.2.2.8
init_logger_for_recording	Sec. 2.2.2.2.9
Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1

Table 2.2-16: logger_init Information.

2.2.2.2.6 init_logger_for_server

This routine initializes the logger for server mode.

Calls	
Function	**Where Described**
logger_init_cmc	Sec. 2.2.2.2.11
logger_message_location	Sec. 2.2.2.5.1
logger_get_remote_rtc	Sec. 2.2.2.2.13
handle_remote_rtc	Sec. 2.2.2.5.13
logger_put_remote_rtc	Sec. 2.2.2.2.14
Called By	
Function	**Where Described**
logger_init	Sec. 2.2.2.2.5

Table 2.2-17: init_logger_for_server Information.

2.2.2.2.7 init_logger_for_copying

This routine initializes the logger for copying. It collects information about files to be copied and opens them.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
start_dt	LoggerTime	p_logger.drn (Appendix A)
start_dstr [DATE_OF_RUN_SZ +1]	char	standard
start_tstr [TIME_OF_RUN_SZ + 1]	char	standard
comment_str [COMMENT_SZ + 1]	char	standard
project_str [PROJECT_ID_SZ + 1]	char	standard
file_nm	pointer to char	standard
ex_ids [loggerMaxExerciseIDs + 1]	char	standard
i	int	standard
j	int	standard
open	BOOL	standard

Errors	
Error	**Reason for Error**
FIO_VERSION1_ERROR	Fio file volume has an invalid version ID (Version1)
FIO_VERSION2_ERROR	Fio file volume has an invalid version ID (Version2)
logger_error	File was specified for both reading and writing

Calls	
Function	**Where Described**
yes_or_no	Sec. 2.2.2.7.9
logger_msg	Sec. 2.2.2.5.2
collect_fiofile_info	Sec. 2.2.2.7.2
open_fio_files	Sec. 2.2.2.2.10
fio_is_null_header	Sec. 2.1.4.2.3.1
get_fileid_fiohdr	Sec. 2.1.4.2.3.12
logger_error	Sec. 2.2.2.5.3
get_datetime	Sec. 2.1.3.2.6
datetime_to_strings	Sec. 2.1.3.2.5
fio_file_header	Sec. 2.1.4.2.3 .3
fio_convert_header	Sec. 2.1.4.2.3.2
fio_print_fiohdr	Sec. 2.1.4.2.3.22

Called By	
Function	**Where Described**
logger_init	Sec. 2.2.2.2.5

Table 2.2-18: init_logger_for_copying Information.

2.2.2.2.8 init_logger_for_playback

This routine initializes the logger for playback. Information is collected about the SIMNET data files to be played back. The network is initialized.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
play_stealth_protocol	BOOL	standard
files	file_info	Sec. 2.2.1.8 util.h

Calls	
Function	**Where Described**
logger_init_cmc	Sec. 2.2.2.2.11
init_file_info	Sec. 2.2.2.8.1
logger_msg	Sec. 2.2.2.5.2
collect_fiofile_info	Sec. 2.2.2.7.2
collect_playback_info	Sec. 2.2.2.7.7
yes_or_no	Sec. 2.2.2.7.9
logger_start	Sec. 2.2.2.2.29

Called By	
Function	**Where Described**
logger_init	Sec. 2.2.2.2.5

Table 2.2-19: init_logger_for_playback Information.

2.2.2.2.9 init_logger_for_recording

This routine initializes the logger for recording. The routine collects information about the CMC Network initialization, the start time of the exercise, and the SIMNET file and file header.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
sim_protocol	BOOL	standard
voice_protocol	BOOL	standard
num_exercises	short	standard
exercises [loggerMaxExerciseIDs+1]	char	standard
std_protocols	short	standard
nstd_protocols	short	standard
start_dt	LoggerTime	p_logger.h
files	file_info	Sec. 2.2.1.8 util.h
comment_str [COMMENT_SZ + 1]	char	standard
project_str [PROJECT_ID_SZ + 1]	char	standard

Calls	
Function	**Where Described**
logger_init_cmc	Sec. 2.2.2.2.11
logger_msg	Sec. 2.2.2.5.2
collect_cmcsubscribe_info	Sec. 2.2.2.7.5
get_datetime	Sec. 2.1.3.2.6
init_file_info	Sec. 2.2.2.8.1
collect_fiofile_info	Sec. 2.2.2.7.2
collect_fioheader_info	Sec. 2.2.2.7.6
yes_or_no	Sec. 2.2.2.7.9
logger_start	Sec. 2.2.2.2.29
Called By	
Function	**Where Described**
init_logger	Sec. 2.2.2.2.5

Table 2.2-20: init_logger_for_recording Information.

2.2.2.2.10 open_fio_files

This routine opens the specified files for the Data Logger.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fls	pointer to file_info	Sec. 2.2.1.8 util.h
fhdr	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	standard
oflags	int	standard
opstr	pointer to char	standard
fd	fio_desc	Sec. 2.1.4.1.1 fiolib.h
Return Values		
Return Value	**Type**	**Meaning**
fd	fio_descr	The file descriptor for the opened file
FIO_ERROR	int	The procedure failed due to an error
Errors		
Error	**Reason for Error**	
FIO_ERROR	No files specified	

Calls	
Function	**Where Described**
logger_error	Sec. 2.2.2.5.3
fio_open	Sec. 2.1.4.2.2.5
logger_msg	Sec. 2.2.2.5.2
fio_concat	Sec. 2.1.4.2.2.6
fio_close	Sec. 2.1.4.2.2.12
fio_error_string	Sec. 2.1.4.2.4.2
Called By	
Function	**Where Described**
logger_start	Sec. 2.2.2.2.29
init_logger_for_copying	Sec. 2.2.2.2.7

Table 2.2-21: open_fio_files Information.

2.2.2.2.11 logger_init_cmc

This routine opens and initializes the CMC network.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cmc_oflags	int	standard
Errors		
Error	**Reason for Error**	
logger_error	- Unable to initialize CMC network - Failed to initialize the CMC filter	
Calls		
Function	**Where Described**	
fio_open	Sec. 2.1.4.2.2.5	
logger_error	Sec. 2.2.2.5.3	
fio_error	Sec. 2.1.4.2.4.1	
cmc_init	Sec. 2.2.2.1.5	
Called By		
Function	**Where Described**	
init_logger_for_server	Sec. 2.2.2.2.6	
init_logger_for_playback	Sec. 2.2.2.2.8	
init_logger_for_recording	Sec. 2.2.2.2.9	

Table 2.2-22: logger_init_cmc Information.

2.2.2.2.12 logger_accept_pdus

This routine sets up the CMC filter. It accepts PDU's of standard protocol (if requested) and non-standard protocol (if requested). Logger run-time commands (RTC) always accepted.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
num_exercises	short	standard
exercises[]	char	standard
num_stdprotocols	short	standard
num_nstdprotocols	short	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	standard
Errors		
Error	**Reason for Error**	
logger_error	- Failed to accept exerciseID - Failed to accept protocols - Failed to reject protocols	
Calls		
Function	**Where Described**	
cmc_accept	Sec. 2.2.2.1.5	
logger_error	Sec. 2.2.2.5.3	
cmc_reject	Sec. 2.2.2.1.4	
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	

Table 2.2-23: logger_accept_pdus Information.

2.2.2.2.13 logger_get_remote_rtc

This routine reads a run time command from the CMC network and places it on the IN command queue.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Errors		
Error	**Reason for Error**	
logger_error	Unexpected fio_read error	
Calls		
Function	**Where Described**	
fio_read	Sec. 2.1.4.2.2.7	
protocol_of	Sec. 2.2.2.5.4	
INC_RCVDPVD_COUNT	Sec. 2.2.1.1 cmc.h (macro definition)	
new_rtc	Sec. 2.2.2.5.14	
F[illegible]_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
IN_QUEUE	Sec. 2.2.1.3 queue.h (macro definition)	

Called By	
Function	**Where Described**
logger_fsm	Sec. 2.2.2.2.15
init_logger_for_server	Sec. 2.2.2.2.6

Table 2.2-24: logger_get_remote_rtc Information.

2.2.2.2.14 logger_put_remote_rtc

This routine reads a run time command from the OUT command queue and places it on the CMC Network.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pktbuf [MAX_FIO_PACKET_SIZE]	char	standard
rtc_available	BOOL	standard
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Errors		
Error	**Reason for Error**	
FIO_NET_WRITE_ERROR	Unable to write packet to network device	
fio_error	Unexpected error on playback	
Calls		
Function	***Where Described***	
queue_count	Sec. 2.2.2.3.8	
dequeue	Sec. 2.2.2.3.7	
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)	
free_rtc	Sec. 2.2.2.5.15	
fio_write	Sec. 2.1.4.2.2.8	
INC_RCVDPVD_COUNT	Sec. 2.2.1.1 cmc.h (macro definition)	
fio_error	Sec. 2.1.4.2.4.1	
Called By		
Function	**Where Described**	
logger_fsm	Sec. 2.2.2.2.15	

Table 2.2-25: logger_put_remote_rtc Information.

2.2.2.2.15 logger_fsm

This routine is the Finite State Machine. The logger spends most of its time here in an internal loop. In each cycle the logger acquires a packet and processes it. For example, during playback, the logger reads a packet from the logger file and writes it to the network. Every so many cycles, the logger checks for a run-time command and processes it if one is found.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cycles	unsigned	standard

Errors	
Error	**Reason for Error**
logger_error	- Failed to set to real-time priority - Running at real-time priority - Unknown activity - Unknown state - Failed to reset to real-time priority - Running at normal priority

Calls	
Function	**Where Described**
logger_copying_state	Sec. 2.2.2.2.16
logger_playback_state	Sec. 2.2.2.2.17
logger_record_state	Sec. 2.2.2.2.19
logger_get_remote_rtc	Sec. 2.2.2.2.13
handle_rtc	Sec. 2.2.2.5.7
logger_put_remote_rtc	Sec. 2.2.2.2.14
logger_clock_tick	Sec. 2.2.2.2.21
logger_error	Sec. 2.2.2.5.3

Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1

Table 2.2-26: logger_fsm Information.

2.2.2.2.16 logger_copying_state

This routine defines a single cycle in the Data Logger's copying state. It reads one packet from the source file and writes it to the destination file.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
done	BOOL	standard
newpkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h

Return Values		
Return Value	**Type**	**Meaning**
loggerConnected		Allows the logger to exit

Errors	
Error	**Reason for Error**
FIO_EOV_ERROR	- Copier finished writing packet # - Copier finished reading packet #
FIO_EOF_ERROR	- Successfully completed copying files - Write files ran out of space
FIO_DRIVE_NOT_READY	Tape drive is not ready for reading
logger_error	Error reading packet
Calls	
Function	**Where Described**
logger_msg	Sec. 2.2.2.5.2
fio_read	Sec. 2.1.4.2.2.7
logger_error	Sec. 2.2.2.5.3
fio_write	Sec. 2.1.4.2.2.8
continue_or_quit	Sec. 2.2.2.7.8
Called By	
Function	**Where Described**
logger_fsm	Sec. 2.2.2.2.15

Table 2.2-27: logger_copying_state Information.

2.2.2.2.17 init_logger_playback_state

This routine initializes state variables used by logger-playbackstate.

Called By	
Function	**Where Described**
logger_continue	Sec. 2.2.2.2.24

Table 2.2-28: init_logger_playback_state Information.

2.2.2.2.18 logger_playback_state

This routine defines a single cycle in the playback state. In the playback state, the Data Logger gets the next packet from the logger file, filtering out certain protocols (i.e. Stealth) if necessary, and attempts to write the packet out to the CMC Network.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
peek	BOOL	standard
peek_pkt	pointer to pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
drive_not_ready_count	int	standard
margin	int	standard
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
protocol_number	int	standard

Return Values		
Return Value	**Type**	**Meaning**
LoggerSuspended	LoggerState	The logger is in the suspended state
logger_state	LoggerState	The current state of the logger
loggerActive	LoggerState	The logger is in the active state

Errors	
Error	**Reason for Error**
FIO_EOV_ERROR	Finished reading packet #
FIO_EOF_ERROR	Ran out of packets during playback; logger is entering the suspended state
FIO_DRIVE_NOT_READY_ERROR	Tape drive is not ready for playback
logger_error	Error reading packet for playback
FIO_NET_WRITE_ERROR	Unexpected error on playback

Calls	
Function	**Where Described**
fio_read	Sec. 2.1.4.2.2.7
logger_msg	Sec. 2.2.2.5.2
logger_suspend	Sec. 2.2.2.2.30
logger_error	Sec. 2.2.2.5.3
fio_error_string	Sec. 2.1.4.2.4.2
protocol_of	Sec. 2.2.2.5.4
on_time	Sec. 2.2.2.6.10
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)
INC_SEND_COUNT	Sec. 2.2.1.1 cmc.h (macro definition)
get_fast_time_factor	Sec. 2.2.2.6.4
INC_LATE_COUNT	Sec. 2.2.1.1 cmc.h (macro definition)
INC_LATE_MARGIN	Sec. 2.2.1.1 cmc.h (macro definition)
INC_DROP_COUNT	Sec. 2.2.1.1 cmc.h (macro definition)
fio_write	Sec. 2.1.4.2.2.8

Called By	
Function	**Where Described**
logger_fsm	Sec. 2.2.2.2.15
logger_continue	Sec. 2.2.2.2.24

Table 2.2-29: logger_playback_state Information.

2.2.2.2.19 logger_record_state

This routine defines a single cycle in the data logger's recording state. It reads one packet from the network. If the packet is a run-time command, it is placed in the RTC queue. Otherwise, it is written out to the logger file.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
retval	int	standard
faile_reads	int	standard
advice_count	int	standard
drive_not_ready_count	int	standard

Return Values		
Return Value	**Type**	**Meaning**
loggerSuspended	LoggerState	The logger is in the suspended state
loggerActive	LoggerState	The logger is in the active state

Errors	
Error	**Reason for Error**
FIO_NO_PACKETS_AVAIL_ERROR	Failing to receive packets
logger_error	- Unexpected fio_read error - Error writing packet during record
FIO_EOF_ERROR	Ran out of space while recording; suspended logger
FIO_DRIVE_NOT_READY_ERROR	Tape drive is not ready for recording. New tape volume required; packets are being lost

Calls	
Function	**Where Described**
fio_read	Sec. 2.1.4.2.2.7
bells	Sec. 2.2.2.2.35
logger_error	Sec. 2.2.2.5.3
print_advice	Sec. 2.2.2.2.37
fio_error_string	Sec. 2.1.4.2.4.2
protocol_of	Sec. 2.2.2.5.4
INC_RCVDPVD_COUNT	Sec. 2.2.1.1 cmc.h (macro definition)
new_rtc	Sec. 2.2.2.5.14
FIO_PACKET_SIZE	Sec. 2.1.4.1.1 fiolib.h (macro definition)
enqueue	Sec. 2.2.2.3.6
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)
fio_write	Sec. 2.1.4.2.2.8
INC_DROP_COUNT	Sec. 2.2.1.1 cmc.h (macro definition)
logger_suspend	Sec. 2.2.2.2.30
logger_msg	Sec. 2.2.2.5.2

Called By	
Function	**Where Described**
logger_fsm	Sec. 2.2.2.2 .15

Table 2.2-30: logger_record_state Information.

2.2.2.2.20 logger_reset

This routine resets the logger, closes all the files, and closes the RACAL if necessary.

Calls	
Function	**Where Described**
fio_close	Sec. 2.1.4.2.2.12
racal_close	Sec. 2.2.2.4.2
logger_message_location	Sec. 2.2.2.5.1
Called By	
Function	**Where Described**
ccatcher	Sec. 2.2.2.2.34
main	Sec. 2.2.2.2.1

Table 2.2-31: logger_reset Information.

2.2.2.2.21 logger_clock_tick

This routine communicates timing information to the client.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
last_clocktick	unsigned long	standard
Calls		
Function	**Where Described**	
send_clock_tick	Sec. 2.2.2.5.29	
Called By		
Function	**Where Described**	
logger_fsm	Sec. 2.2.2.2.15	

Table 2.2-32: logger_clock_tick Information.

2.2.2.2.22 logger_op_error

This routine performs a preliminary check for errors in the logger operation. It makes sure that the operation requested by the user is meaningful.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
op	LoggerRTCType	/simnet/common/include/protocol/p_logger.h

Returns
Returns pointer to char
"Logger is not connected"
"Logger has already been started"
"Logger is already in active mode"
"Can't activate logger which has not been started"
"Can't suspend the logger while loggerCopying
"Logger has already been suspended"
"Can't suspend logger which has not been started"
"Seeking is allowed during loggerPlayback only"
"Changing speed is allowed during loggerPlayback only"
"Cannot change logger speed when using the Racal"
"Invalid logger runtime command"
NULL

Called By	
Function	**Where Described**
command_request_handler	Sec. 2.2.2.5.23
handle_stdin_handler	Sec. 2.2.2.5.8

Table 2.2-33: logger_op_error Information.

2.2.2.2.23 logger_connect

This procedure changes the logger's state to logger connected, meaning that the logger has connected to a client in server mode. This procedure takes no parameters, has no internal variables, returns no values, and has no errors.

Called By	
Function	**Where Described**
connect_request_handler	Sec. 2.2.2.5.21

Table 2.2-34: logger_connect Information.

2.2.2.2.24 logger_continue

This routine changes the logger's state to logger active, meaning the logger will continue to process packets. It also restarts the exercise clock if the logger is playing back.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
pkt	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
et	int	standard
ct	int	standard

Calls	
Function	**Where Described**
logger_msg	Sec. 2.2.2.5.2
unfreeze_time	Sec. 2.2.2.6.9
exe_to_cmc_time	Sec. 2.2.2.6.1
racal_seek_to_time	Sec. 2.2.2.4.12
init_logger_playback_state	Sec. 2.2.2.2.17
logger_playback_state	Sec. 2.2.2.2.18
set_time	Sec. 2.2.2.6.6
FIO_PACKET_TIME	Sec. 2.1.4.1.1 fiolib.h (macro definition)
get_time	Sec. 2.2.2.6.7
fio_flush	Sec. 2.1.4.2.2.11
cmc_zero_status	Sec. 2.2.2.1.7
Called By	
Function	**Where Described**
handle_stdin_rtc	Sec. 2.2.2.5.8
command_request_handler	Sec. 2.2.2.5.23

Table 2.2-35: logger_continue Information.

2.2.2.2.25 logger_disconnect

This routine changes the logger's state to logger disconnected, meaning the logger is no longer connected to its client.

Calls	
Function	**Where Described**
logger_stop	Sec. 2.2.2.2.28
disconnect_handler	Sec. 2.2.2.5.22

Table 2.2-36: logger_disconnect Information.

2.2.2.2.26 logger_seek_relative

This routine sets the playback file pointer relative to its current location based on "rtime" and the current packet's timestamp.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rtime	int	standard
relative_current_location	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
cur_time	int	standard
Errors		
Error	**Reason for Error**	
FIO_ERROR	Fio_seek failed	

Calls	
Function	**Where Described**
fio_flush	Sec. 2.1.4.2.2.11
fio_seek	Sec. 2.1.4.2.2.9
logger_error	Sec. 2.2.2.5.3
set_time	Sec. 2.2.2.6.6
racal_seek_to_time	Sec. 2.2.2.4.12
racal_stop	Sec. 2.2.2.4.11
logger_msg	Sec. 2.2.2.5.2
Called By	
Function	**Where Described**
goto_handler	Sec. 2.2.2.5.9
scan_handler	Sec. 2.2.2.5.12
command_request_handler	Sec. 2.2.2.5.23

Table 2.2-37: logger_seek_relative Information.

2.2.2.2.27 logger_speed

This routine changes the playback speed of the logger. The playback speed is set to the factor of the parameter *factor*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
factor	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ft	int	standard
Calls		
Function	**Where Described**	
set_fast_time_factor	Sec. 2.2.2.6.3	
logger_msg	Sec. 2.2.2.5.2	
time_string	Sec. 2.2.2.6.11	
Called By		
Function	**Where Described**	
handle_stdin_rtc	Sec. 2.2.2.5.8	
command_request_handler	Sec. 2.2.2.5.23	

Table 2.2-38: logger_speed Information.

2.2.2.2.28 logger_stop

This routine stops the logger and places the finite state machine into the loggerConnected state.

Calls	
Function	**Where Described**
cmc_report_status	Sec. 2.2.2.1.8
get_time	Sec. 2.2.2.6.7
Called By	
Function	**Where Described**
logger_disconnect	Sec. 2.2.2.2.25
handle_stdin_rtc	Sec. 2.2.2.5.8
command_request_handler	Sec. 2.2.2.5.23

Table 2.2-39: logger_stop Information.

2.2.2.2.29 logger_start

This routine starts the Data Logger by opening the files for reading and writing, and getting the start time for the exercise and passing it back to the client. Finally, it places the logger in logger suspended state.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
files	pointer to file_info	Sec. 2.2.1.8 util.h
start_dt	pointer to LoggerTime	p_logger.h
recording	BOOL	standard
projid	pointer to char	standard
comment	pointer to char	standard
is_racal	BOOL	standard
num_exercises	short	standard
exercises	pointer to char	standard
num_stdprotocols	short	standard
num_nstdprotocols	short	standard
record_voice	BOOL	standard
record_sim	BOOL	standard
play_stealth	BOOL	standard
Return Values		
Return Value	**Type**	**Meaning**
FALSE	BOOL	The procedure failed
TRUE	BOOL	The procedure succeeded; the logger is started
Errors		
Error	**Reason for Error**	
fio_error	- Failed to open files for loggerRecording - Failed to open files for loggerPlayback	

Calls	
Function	**Where Described**
cmc_protocol_types	Sec. 2.2.2.1.2
logger_accept_pdus	Sec. 2.2.2.2.12
datetime_to_strings	Sec. 2.1.3.2.5
fio_file_header	Sec. 2.1.4.2.3.3
open_fio_files	Sec. 2.2.2.2.10
logger_error	Sec. 2.2.2.5.3
racal_open	Sec. 2.2.2.4.1
collect_racal_time_info	Sec. 2.2.2.7.1
reset_racal_time	Sec. 2.2.2.4.8
racal_start_recording	Sec. 2.2.2.4.9
get_start_date_fiohdr	Sec. 2.1.4.2.3.15
get_start_time_fiohdr	Sec. 2.1.4.2.3.19
strings_to_datetime	Sec. 2.1.3.2.4
fio_print_fiohdr	Sec. 2.1.4.2.3.22
reset_time	Sec. 2.2.2.6.5
fio_flush	Sec. 2.1.4.2.2.11
logger_suspend	Sec. 2.2.2.2.30
Called By	
Function	**Where Described**
init_logger_for_recording	Sec. 2.2.2.2.9
init_logger_for_playback	Sec. 2.2.2.2.8
command_request_handler	Sec. 2.2.2.5.23

Table 2.2-40: logger_start Information.

2.2.2.2.30 logger_suspend

This routine places the finite state machine into the loggerSuspended state.

Calls	
Function	**Where Described**
bells	Sec. 2.2.2.2.35
logger_msg	Sec. 2.2.2.5.2
freeze_time	Sec. 2.2.2.6.8
racal_stop	Sec. 2.2.2.4.11
time_string	Sec. 2.2.2.6.11
get_time	Sec. 2.2.2.6.7
Called By	
Function	**Where Described**
logger_start	Sec. 2.2.2.2.29
logger_playback_state	Sec. 2.2.2.2.18
logger_record_state	Sec. 2.2.2.2.19
handler_stdin_rtc	Sec. 2.2.2.5.8
command_request_handler	Sec. 2.2.2.5.23

Table 2.2-41: logger_suspend Information.

2.2.2.2.31 logger_status

This routine passes out information about the Data Logger's status.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
activity	pointer to LoggerActivity	p_logger.h
operation	pointer to LoggerOperation	p_logger.h
state	pointer to LoggerState	p_logger.h
is_racal	pointer to int	standard
speed	pointer to int	standard
start	pointer to LoggerTime	p_logger.h
offset	pointer to int	standard
packets	pointer to int	standard
file	pointer to pointer to char	standard
project	pointer to pointer to char	standard
comment	pointer to pointer to char	standard
fio_head	pointer to pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ofs	double	standard

Calls	
Function	**Where Described**
get_fast_time_factor	Sec. 2.2.2.6.4
get_time	Sec. 2.2.2.6.7
do_status_reply	Sec. 2.2.2.5.28

Table 2.2-42: logger_status Information.

2.2.2.2.32 logger_get_state

This routine gets the current state of the Data Logger Finite State Machine.

Return Values		
Return Value	**Type**	**Meaning**
logger_state		The current state of the Logger Finite State Machine

Called By	
Function	**Where Described**
avail_request_handler	Sec. 2.2.2.5.20

Table 2.2-43: logger_get_state Information.

2.2.2.2.33 logger_get_activity

This routine gets the current activity of the Data Logger.

Return Values		
Return Value	**Type**	**Meaning**
logger_activity		The current activity of the Logger

Called By	
Function	**Where Described**
help_handler	Sec. 2.2.2.5.10

Table 2.2-44: logger_get_activity Information.

2.2.2.2.34 ccatcher

This routine ontrols operation of the Data Logger signal interrupt handler.

Calls	
Function	**Where Described**
logger_msg	Sec. 2.2.2.5.2
cmc_report_status	Sec. 2.2.2.1.8
get_time	Sec. 2.2.2.6.7
logger_reset	Sec. 2.2.2.2.20

Table 2.2-45: ccatcher Information.

2.2.2.2.35 bells

This routine prints bell sounds.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
i	int	standard

Called By	
Function	**Where Described**
logger_record_state	Sec. 2.2.2.2.19
logger_suspend	Sec. 2.2.2.2.30

Table 2.2-46: bells Information.

2.2.2.2.36 print_command_line_help

This routine prints help messages:

Must specify one of the following flags:
-c(opy or convert data files)
-p(lay tape of net traffic)
-r(ecord net traffic)

-h(elp)
-s(erver mode standard control from PVD)
-v(ersion information)

-q(uery the full set of options)

Parameters		
Parameter	**Type**	**Where Typedef Declared**
progname	pointer to char	standard
Calls		
Function	**Where Described**	
logger_msg	Sec. 2.2.2.5.2	
Called By		
Function	**Where Described**	
process_parms	Sec. 2.2.2.2.2	

Table 2.2-47: print_command_line_help Information.

2.2.2.2.37 print_advice

This routine prints the following advice as necessary:

There are no vehicles on the net. If so, no problem.
The Data Analysis net has become disconnected from the simulation network. Check the transceiver cable connections.
The CMC card has died. Login as root
Kill ringstart and then run netstart.

Calls	
Function	**Where Described**
logger_msg	Sec. 2.2.2.5.2
Called By	
Function	**Where Described**
logger_record_state	Sec. 2.2.2.2.19

Table 2.2-48: print_advice Information.

2.2.2.3 queue.c

This file handles the Run-Time Command Queues when the Data Logger is in server mode.

Includes:
<stdio.h>
"queue.h"

External procedure declarations:
malloc()

Typedefs:

queue_elt_type	QUEUE_ELT
queue_type	QUEUE

Global Variables:

QUEUE_ELT	free_elts
QUEUE	in_queue
QUEUE	out_queue

2.2.2.3.1 queue_of

This routine returns the address of the free_queue, out_queue, or in_queue for the specified queue_id.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
queue_id	int	standard
Return Values		
Return Value	**Type**	**Meaning**
&free_queue	pointer to QUEUE	The address of free_queue
&out_queue	pointer to QUEUE	The address of out_queue
&in_queue	pointer to QUEUE	The address of in_queue
NULL	pointer to QUEUE	Invalid queue_id
Called By		
Function	**Where Described**	
enqueue	Sec. 2.2.2.3.6	
dequeue	Sec. 2.2.2.3.7	
queue_count	Sec. 2.2.2.3.8	

Table 2.2-49: queue_of Information.

2.2.2.3.2 new_queue_elt

This routine creates a new queue element.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
qelt	pointer to QUEUE_ELT	Sec. 2.2.2.3 queue.c
Return Values		
Return Value	**Type**	**Meaning**
qelt	pointer to QUEUE_ELT	pointer to the new queue element
Called By		
Function	**Where Described**	
enqueue	Sec. 2.2.2.3.6	

Table 2.2-50: new_queue_elt Information.

2.2.2.3.3 free_queue_elt

This routine frees a queue element.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
qelt	pointer to QUEUE_ELT	Sec. 2.2.2.3 queue.c
Called By		
Function	**Where Described**	
dequeue	Sec. 2.2.2.3.7	

Table 2.2-51: free_queue_elt Information.

2.2.2.3.4 init_queues

This routine initializes the queues with the following characteristics:

free_elts = NULL

free_queue.count = 0
free_queue.head = NULL
free_queue.tail = NULL

out_queue.count = 0
out_queue.head = NULL
out_queue.tail = NULL

in_queue.count = 0
in_queue.head = NULL
in_queue.tail = NULL

2.2.2.3.5 reinit_queues

This routine reinitializes the queues.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ptr	pointer to char	standard
Calls		
Function	**Where Described**	
queue_count	Sec. 2.2.2.3.8	
dequeue	Sec. 2.2.2.3.7	
enqueue	Sec. 2.2.2.3.6	
Called By		
Function	**Where Described**	
main	Sec. 2.2.2.2.1	
logger_init	Sec. 2.2.2.2.5	

Table 2.2-52: reinit_queues Information.

2.2.2.3.6 enqueue

Ths routine makes an entry to the specified queue.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
queue	int	standard
ptr	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
qptr	pointer to QUEUE	Sec. 2.2.2.3 queue.c
qelt	pointer to QUEUE_ELT	Sec. 2.2.2.3 queue.c
Calls		
Function		**Where Described**
queue_of		Sec. 2.2.2.3.8
new_queue_elt		Sec. 2.2.2.3.2
Called By		
Function		**Where Described**
logger_record_state		Sec. 2.2.2.2.19
send_rtc		Sec. 2.2.2.5.17
free_rtc		Sec. 2.2.2.5.15

Table 2.2-53: enqueue Information.

2.2.2.3.7 dequeue

This routine removes an entry from the specified queue.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
queue	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
qptr	pointer to QUEUE	Sec. 2.2.2.3 queue.c
qelt	pointer to QUEUE_ELT	Sec. 2.2.2.3 queue.c
ptr	pointer to char	standard
Return Values		
Return Value	**Type**	**Meaning**
ptr	pointer to char	
Calls		
Function		**Where Described**
queue_of		Sec. 2.2.2.3.8
free_queue_elt		Sec. 2.2.2.3.3

Called By	
Function	**Where Described**
logger_put_remote_rtc	Sec. 2.2.2.2.14
new_rtc	Sec. 2.2.2.5.14
receive_rtc	Sec. 2.2.2.5.16

Table 2.2-54: dequeue Information.

2.2.2.3.8 queue_count

This routine returns the queue count of the specified queue.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
queue	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
qptr	pointer to QUEUE	Sec. 2.2.2.3 queue.c
Return Values		
Return Value	**Type**	**Meaning**
qptr->count	int	The queue count
Calls		
Function	**Where Described**	
queue_of	Sec. 2.2.2.3.1	
Called By		
Function	**Where Described**	
logger_put_remote_rtc	Sec. 2.2.2.2.14	
new_rtc	Sec. 2.2.2.5.14	
receive_rtc	Sec. 2.2.2.5.16	

Table 2.2-55: queue_count Information.

2.2.2.4 racal.c

This file provides the interface to the Racal audio recorder for recording the voice traffic received over CB radios. This capability has become obsolete with the advent of digital voice recording where voice traffic is digitized and then transmitted over the SIMNET network the same as all other vehicle simulator information.

Includes:

```
<sys/ioctl.h>
<stdio.h>
<fnctl.h>
<termio.h>
<ctype.h>
"cmc.h"
"racal.h"
```

Standard Procedures:

perror()
exit()
sleep()
strcat()

Defines:

TERMINAL_LINE
SEC_FIELD
MIN_FIELD
HOUR_FIELD
DAY_FIELD

File descriptor for terminal line to racal recorder declaration:

racal_fd

2.2.2.4.1 racal_open

This routine opens the racal, setting the line's baud rate to 4800 baud.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
play	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
open_mask	int	standard
dev_name	pointer to char	standard
att_term_buffer	struct termio	standard
Errors		
Error	**Reason for Error**	
perror	- Failed to read terminal characteristics for device - Failed to read ATT style terminal characteristics - Failed to set output AON/XOFF control for device	
Calls		
Function	**Where Described**	
send_to_racal	Sec. 2.2.2.4.4	
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	

Table 2.2-56: racal_open Information.

2.2.2.4.2 racal_close

This routine closes the racal.

Called By	
Function	**Where Described**
logger_reset	Sec. 2.2.2.2.20

Table 2.2-57: racal_close Information.

2.2.2.4.3 get_from_racal

This routine reads and returns a terminated line of text from the racal recorder. For efficiency, the NO_DELAY is open and reply is accumulated over several calls.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
in_char	char	standard
i	int	standard
reply[264]	char	standard
reply_ptr	pointer to char	standard
Return Values		
Return Value	**Type**	**Meaning**
reply_ptr	pointer to char	terminated line of text from the racal recorder
Called By		
Function	**Where Described**	
get_racal_time	Sec. 2.2.2.4.7	

Table 2.2-58: get_from_racal Information.

2.2.2.4.4 send_to_racal

This routine sends a command string to the racal recorder open on *fd*.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
fd	int	standard
command_string	pointer to char	standard

Called By	
Function	**Where Described**
racal_open	Sec. 2.2.2.4.1
get_racal_time	Sec. 2.2.2.4.7
reset_racal_time	Sec. 2.2.2.4.8
racal_start_recording	Sec. 2.2.2.4.9
racal_start_playing	Sec. 2.2.2.4.10
racal_stop	Sec. 2.2.2.4.11
racal_seek_to_time	Sec. 2.2.2.4.12

Table 2.2-59: send_to_racal Information.

2.2.2.4.5 racal_to_cmc

This routine accepts the time string returned by the racal and returns the corresponding time in cmc milliseconds. The string format is "001 RTyymmddhhmmss\015". Note that strings in the form "001 A\015" may need weeding out.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
time_string	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
second	int	standard
minute	int	standard
hour	int	standard
day	int	standard
cmc_msec	int	standard
Return Values		
Return Value	**Type**	**Meaning**
cmc_msec	int	*time_string* in corresponding cmc milliseconds
Called By		
Function	**Where Described**	
get_racal_time	Sec. 2.2.2.4.7	

Table 2.2-60: racal_to_cmc Information.

2.2.2.4.6 seconds_to_racal_string

This routine converts passed integer seconds into a string in the form yymmddhhmmss. Note that this routine will handle no more than 24 hours, and that the returned string will be destroyed on the next call to this routine.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
seconds	int	standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
time_string[14]	char	standard
years	int	standard
months	int	standard
days	int	standard
hours	int	standard
minutes	int	standard
Return Values		
Return Value	**Type**	**Meaning**
time_string	char	conversion of seconds into a time string.
Called By		
Function	**Where Described**	
racal_seek_to_time	Sec. 2.2.2.4.12	

Table 2.2-61: seconds_to_racal_string Information.

2.2.2.4.7 get_racal_time

This routine returns immediately with the time in milliseconds.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
reply	pointer to char	standard
Return Values		
Return Value	**Type**	**Meaning**
racal_to_cmc(reply)	int	time in milliseconds
Calls		
Function	**Where Described**	
send_to_racal	Sec. 2.2.2.4.4	
get_from_racal	Sec. 2.2.2.4.3	
racal_to_cmc	Sec. 2.2.2.4.5	
Called By		
Function	**Where Described**	
racal_seek_to_time	Sec. 2.2.2.4.12	

Table 2.2-62: get_racal_time Information.

2.2.2.4.8 reset_racal_time

This routine sets the racal's real-time clock to 0. Note that the racal's clock continues to run.

Calls	
Function	**Where Described**
send_to_racal	Sec. 2.2.2.4.4
Called By	
Function	**Where Described**
logger_start	Sec. 2.2.2.2.29

Table 2.2-63: reset_racal_time Information.

2.2.2.4.9 racal_start_recording

This routine starts the racal recording.

Calls	
Function	**Where Described**
send_to_racal	Sec. 2.2.2.4.4

Table 2.2-64: racal_start_recording Information.

2.2.2.4.10 racal_start_playing

This routine starts the racal playing.

Calls	
Function	**Where Described**
send_to_racal	Sec. 2.2.2.4.4

Table 2.2-65: racal_start_playing Information.

2.2.2.4.11 racal_stop

This routine stops the racal.

Calls	
Function	**Where Described**
send_to_racal	Sec. 2.2.2.4.4
Called By	
Function	**Where Described**
logger_suspend	Sec. 2.2.2.2.30
logger_seek_relative	Sec. 2.2.2.2.26

Table 2.2-66: racal_stop Information.

2.2.2.4.12 racal_seek_to_time

This routine searches and waits until racal time exceeds passed time. Note that this assumes that CMC time is not advancing; CMC time is in Freeze state.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
seek_time	int	standard
racal_time_at_start	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
offset	int	standard
logger_time	int	standard
current_racal_time	int	standard
old_racal_time	int	standard
racal_time_string	pointer to char	standard
racal_command[200]	char	standard
seconds_to_racal_string()	pointer to char	standard
Calls		
Function	**Where Described**	
seconds_to_racal_string	Sec. 2.2.2.4.6	
send_to_racal	Sec. 2.2.2.4.4	
get_racal_time	Sec. 2.2.2.4.7	
Called By		
Function	**Where Described**	
logger_seek_relative	Sec. 2.2.2.2.26	
logger_continue	Sec. 2.2.2.2.24	

Table 2.2-67: racal_seek_to_time Information.

2.2.2.5 rtc.c

This file handles the processing of run-time commands from the user for both the stand-alone Data Logger and the server Data Logger.

Includes:

<stdio.h>
<sys/ioctl.h>
"fiolib.h"
"fioheader.h"
"network.h"
"p_logger.h"
"p_num.h"
"p_assoc.h"
"p_p2p.h"
"global.h"
"logger.h"
"queue.h"
"rtc.h"
"timer.h"
"util.h"

Standard Procedure Declarations:

malloc()
strncpy()

Static Procedure Declarations:

handle_stdin_rtc()
goto_handler()
help_handler()
info_handler()
scan_handler()
receive_rtc()
send_rtc()
init_pdu()
init_apdu()
avail_request_handler()
connect_request_handler()
disconnect_handler()
command_request_handler()
parse_start_file_info()
status_request_handler()
do_information()
do_status_reply

Defines:

CMC_MSECONDS_PER_SECOND
PDUKind()
STRING_LENGTH

Global Variable Declarations:

loggerMCA
logger_is_connected
clientSimAddr
loggerSimAddr
msgbuf[256]
send_to_network
logger_jump_table[loggerLastPDUKind]

Typedef:
jump_struct JUMP_ENTRY

2.2.2.5.1 logger_message_location

This routine determines the place to send logger messages: stdout/stderr or network.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msg_to_net	BOOL	standard
Called By		
Function	**Where Described**	
main	Sec. 2.2.2.2.1	
logger_reset	Sec. 2.2.2.2.20	
init_logger_for_server	Sec. 2.2.2.2.6	

Table 2.2-68: logger_message_location Information.

2.2.2.5.2 logger_msg

This routine sends a message to the user.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msg	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
Calls		
Function	**Where Described**	
new_rtc	Sec. 2.2.2.5.14	
do_information	Sec. 2.2.2.5.26	
init_apdu	Sec. 2.2.2.5.18	
send_rtc	Sec. 2.2.2.5.17	

Called By	
Function	**Where Described**
cmc_reprot_status	Sec. 2.2.2.1.8
main	Sec. 2.2.2.2.1
process_parms	Sec. 2.2.2.2.2
logger_suspend	Sec. 2.2.2.2.30
print_command_line_help	Sec. 2.2.2.2.36
print_advice	Sec. 2.2.2.2.37
ccatcher	Sec. 2.2.2.2.34
logger_seek_relative	Sec. 2.2.2.2.26
logger_speed	Sec. 2.2.2.2.27
logger_playback_state	Sec. 2.2.2.2.18
logger_record_state	Sec. 2.2.2.2.19
logger_continue	Sec. 2.2.2.2.24
logger_copying_state	Sec. 2.2.2.2.16
init_logger_for_playback	Sec. 2.2.2.2.8
init_logger_for_recording	Sec. 2.2.2.2.9
open_fio_files	Sec. 2.2.2.2.10
init_logger_for_copying	Sec. 2.2.2.2.7
handle_stdin_rtc	Sec. 2.2.2.5.8
help_handler	Sec. 2.2.2.5.10
handle_remote_rtc	Sec. 2.2.2.5.13

Table 2.2-69: logger_msg Information.

2.2.2.5.3 logger_error

This routine sends an error message to the user.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msg	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
Calls		
Function	**Where Described**	
new_rtc	Sec. 2.2.2.5.14	
do_information	Sec. 2.2.2.5.26	
init_apdu	Sec. 2.2.2.5.18	
send_rtc	Sec. 2.2.2.5.17	

Called By	
Function	**Where Described**
cmc_accept	Sec. 2.2.2.1.3
cmc_reject	Sec. 2.2.2.1.4
process_parms	Sec. 2.2.2.2.2
logger_seek_relative	Sec. 2.2.2.2.26
logger_start	Sec. 2.2.2.2.29
logger_playback_state	Sec. 2.2.2.2.18
logger_record_state	Sec. 2.2.2.2.19
logger_accept_pdus	Sec. 2.2.2.2.12
logger_fsm	Sec. 2.2.2.2.15
logger_copying_state	Sec. 2.2.2.2.16
open_fio_files	Sec. 2.2.2.2.10
init_logger_for_copying	Sec. 2.2.2.2.7
handle_stdin_rtc	Sec. 2.2.2.5.8
command_request_handler	Sec. 2.2.2.5.23

Table 2.2-70: logger_error Information.

2.2.2.5.4 protocol_of

This routine returns the specified fio_packet's protocol

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
get_root_protocol	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to PointToPointPDU	p_p2p.h
protocol_number	AssociationUserProtocol	p_assoc.h
Return Values		
Return Value	**Type**	**Meaning**
protocol_number	AssociationUserProtocol	the protocol of the packet
Called By		
Function	**Where Described**	
logger_record_state	Sec. 2.2.2.2.19	
logger_playback_state	Sec. 2.2.2.2.18	
logger_get_remote_rtc	Sec. 2.2.2.2.13	

Table 2.2-71: protocol_of Information.

2.2.2.5.5 rtc_size

This routine returns the size of the specified RTC APDU.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
apdu	pointer to AssociationPDU	p_assoc.h
Return Values		
Return Value	**Type**	**Meaning**
sizeof(AssociationPDU + (unsigned) apdu-> dataLength)	unsigned	the size of the APDU
Called By		
Function	**Where Described**	
send_rtc	Sec. 2.2.2.5.17	

Table 2.2-72: rtc_size Information.

2.2.2.5.6 init_rtc

This routine initializes the rtc module.

Calls	
Function	**Where Described**
AssocCreateMCA	Sec. 2.20.1 See MCC CSCI SDD
AssocGetSimAddress	Sec. 2.20.1 See MCC CSCI SDD
Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1

Table 2.2-73: init_rtc Information.

2.2.2.5.7 handle_rtc

This routine handles processing of input from the user, either from stdin or the network, during execution of the logger.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
is_server	BOOL	standard
Calls		
Function	**Where Described**	
handle_remote_rtc	Sec. 2.2.2.5.13	
handle_stdin_rtc	Sec. 2.2.2.5.8	

Called By	
Function	**Where Described**
logger_fsm	Sec. 2.2.2.2.15

Table 2.2-74: handle_rtc Information.

2.2.2.5.8 handle_stdin_rtc

This routine handles processing of runtime messages from stdin (*i.e.*, the user) when the logger is in command-line mode.

c - Continue
f - Forward scan
g - Goto
h - Help
i - Information about logger status
p - Pause
r - Reverse scan
s - Speed

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
input[STRING_LENGTH + 1]	char	standard
next_char	char	standard
msg	pointer to char	standard
readfds	int	standard
writefds	int	standard

Errors	
Error	**Reason for Error**
logger_error	Unrecognized command sequence

Calls	
Function	**Where Described**
logger_op_error	Sec. 2.2.2.2.22
logger_error	Sec. 2.2.2.5.3
logger_msg	Sec. 2.2.2.5.2
logger_continue	Sec. 2.2.2.2.24
scan_handler	Sec. 2.2.2.5.12
goto_handler	Sec. 2.2.2.5.9
help_handler	Sec. 2.2.2.5.10
info_handler	Sec. 2.2.2.5.11
logger_stop	Sec. 2.2.2.2.28
logger_suspend	Sec. 2.2.2.2.30
logger_speed	Sec. 2.2.2.2.27

Called By	
Function	**Where Described**
handle_rtc	Sec. 2.2.2.5.7

Table 2.2-75: handle_stdin_rtc Information.

2.2.2.5.9 goto_handler

This routine handles processing of Goto runtime commands.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
input	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
c	char	standard
buf[32]	char	standard
seconds	int	standard
dseconds	double	standard
Calls		
Function	**Where Described**	
logger_seek_relative	Sec. 2.2.2.2.26	
Called By		
Function	**Where Described**	
handle_stdin_rtc	Sec. 2.2.2.5.8	

Table 2.2-76: goto_handler Information.

2.2.2.5.10 help_handler

This routine handles processing of Help runtime commands.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
input	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str	pointer to char	standard
activity	int	standard
Calls		
Function	**Where Described**	
logger_get_activity	Sec. 2.2.2.2.33	
logger_msg	Sec. 2.2.2.5.2	
Called By		
Function	**Where Described**	
handle_stdin_rtc	Sec. 2.2.2.5.8	

Table 2.2-77: help_handler Information.

2.2.2.5.11 info_handler

This routine handles processing of Info runtime commands.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
input	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
activity	LoggerActivity	p_logger.h
operation	LoggerOperation	p_logger.h
state	LoggerState	p_logger.h
is_racal	int	standard
speed	int	standard
start	LoggerTime	p_logger.h
offset	int	standard
packets	int	standard
file	pointer to char	standard
project	pointer to char	standard
comment	pointer to char	standard
fio_head	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h
Called By		
Function	**Where Described**	
handle_stdin_rtc	Sec. 2.2.2.5.8	

Table 2.2-78: info_handler Information.

2.2.2.5.12 scan_handler

This routine handles processing of Forward and Reverse Scan runtime commands.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
input	pointer to char	standard
forward	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
c	char	standard
buf[32]	char	standard
seconds	int	standard
dseconds	double	standard
Calls		
Function	**Where Described**	
string_to_seconds	Sec. 2.1.3.2.3	
logger_seek_relative	Sec. 2.2.2.2.26	

Called By	
Function	**Where Described**
handle_stdin_rtc	Sec. 2.2.2.5.8

Table 2.2-79: scan_handler Information.

2.2.2.5.13 handle_remote_rtc

This routine handles processing of runtime messages from the client when the logger is in server mode.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
(fnc) ()	void	standard
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
Calls		
Function	**Where Described**	
receive_rtc	Sec. 2.2.2.5.16	
logger_msg	Sec. 2.2.2.5.2	
free_rtc	Sec. 2.2.2.5.15	
Called By		
Function	**Where Described**	
init_logger_for_server	Sec. 2.2.2.2.6	
handle_rtc	Sec. 2.2.2.5.7	

Table 2.2-80: handle_remote_rtc Information.

2.2.2.5.14 new_rtc

This routine gets a new runtime commandbuffer.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Return Values		
Return Value	**Type**	**Meaning**
rtc	pointer to fio_packet	The new runtime command
Calls		
Function	**Where Described**	
queue_count	Sec. 2.2.2.3.8	
dequeue	Sec. 2.2.2.3.7	

Called By	
Function	**Where Described**
logger_record_state	Sec. 2.2.2.2.19
logger_get_remote_rtc	Sec. 2.2.2.2.13
logger_msg	Sec. 2.2.2.5.2
logger_error	Sec. 2.2.2.5.3
avail_request_handler	Sec. 2.2.2.5.20
connect_request_handler	Sec. 2.2.2.5.21
disconnect_handler	Sec. 2.2.2.5.22
command_request_handler	Sec. 2.2.2.5.23
status_request_handler	Sec. 2.2.2.5.25
send_clock_tick	Sec. 2.2.2.5 .29

Table 2.2-81: new_rtc Information.

2.2.2.5.15 free_rtc

This routine frees a runtime commandbuffer.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Calls		
Function	**Where Described**	
enqueue	Sec. 2.2.2.3.6	
Called By		
Function	**Where Described**	
logger_put_remote_rtc	Sec. 2.2.2.2.14	
handle_remote_rtc	Sec. 2.2.2.5.13	

Table 2.2-82: free_rtc Information.

2.2.2.5.16 receive_rtc

This routine gets the next runtime command, if there is one, from the IN_QUEUE.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Return Values		
Return Value	**Type**	**Meaning**
rtc	pointer to fio_packet	the next runtime command
Calls		
Function	**Where Described**	
queue_count	Sec. 2.2.2.3.8	
dequeue	Sec. 2.2.2.3.7	

Called By	
Function	**Where Described**
handle_remote_rtc	Sec. 2.2.2.5.13

Table 2.2-83: receive_rtc Information.

2.2.2.5.17 send_rtc

This routine places the rtc response in the OUT_QUEUE so it can be written out on the net at the next available moment.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Calls		
Function	**Where Described**	
rtc_size	Sec. 2.2.2.5.5	
enqueue	Sec. 2.2.2.3.6	
Called By		
Function	**Where Described**	
logger_msg	Sec. 2.2.2.5.2	
logger_error	Sec. 2.2.2.5.3	
avail_request_handler	Sec. 2.2.2.5.20	
connect_request_handler	Sec. 2.2.2.5.21	
disconnect_handler	Sec. 2.2.2.5.22	
command_request_handler	Sec. 2.2.2.5.23	
status_request_handler	Sec. 2.2.2.5.25	
send_clock_tick	Sec. 2.2.2.5.29	

Table 2.2-84: send_rtc Information.

The following routines (2.2.2.5.18 through 2.2.2.5.29) handle run-time requests and replies from the client when the logger is in server mode. Run-time fio_packets have the following structure:

fio_packet_header
APDU header
Logger PDU

2.2.2.5.18 init_apdu

This routine initializes the APDU header portion of the run-time command fio_packets. All Logger PDU's are sent via Association datagram PDU's. The following characteristics will be constant for all Logger/Client PDU's.

version: loggerProtocolVersionCurrent
kind: datagramAPDUKind
dataLength: data_len
userProtocol: loggerProtocolNumber

Parameters		
Parameter	**Type**	**Where Typedef Declared**
apdu	pointer to AssociationPDU	p_assoc.h
data_len	unsigned	standard
Called By		
Function	**Where Described**	
logger_msg	Sec. 2.2.2.5.2	
logger_error	Sec. 2.2.2.5.3	
connect_request_handler	Sec. 2.2.2.5.21	
disconnect_handler	Sec. 2.2.2.5.22	
command_request_handler	Sec. 2.2.2.5.23	
status_request_handler	Sec. 2.2.2.5.25	
send_clock_tick	Sec. 2.2.2.5.29	
do_information	Sec. 2.2.2.5.26	

Table 2.2-85: init_apdu Information.

2.2.2.5.19 init_pdu

This routine initializes the Logger PDU's portion of teh runtime command fio_packets with the following characteristics:

version: loggerProtocolVersionCurrent
kind: kind

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pdu	pointer to LoggerPDU	p_logger.h
destination	pointer to SimulationAddress	address.h
kind	int	standard
Called By		
Function	**Where Described**	
avail_request_handler	Sec. 2.2.2.5.20	
connect_request_handler	Sec. 2.2.2.5.21	
disconnect_handler	Sec. 2.2.2.5.22	
command_request_handler	Sec. 2.2.2.5.23	
do_clock_tick	Sec. 2.2.2.5.27	
do_status_reply	Sec. 2.2.2.5.28	

Table 2.2-86: init_pdu Information.

2.2.2.5.20 avail_request_handler

This routine handles AvailRequest PDU's. The client sends out an AvailRequest PDU to find out which Data Loggers are available for connection in server mode. If available, the Data Logger must respond with an AvailReply PDU.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
inrtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
Calls		
Function	**Where Described**	
new_rtc	Sec. 2.2.2.5.14	
logger_get_state	Sec. 2.2.2.2.32	
init_pdu	Sec. 2.2.2.5.19	
init_apdu	Sec. 2.2.2.5.18	
send_rtc	Sec. 2.2.2.5.17	

Table 2.2-87: avail_request_handler Information.

2.2.2.5.21 connect_request_handler

This routine handles ConnectRequest PDU's. The client sends out an ConnectRequest PDU to the particular Data Logger it wishes to connect to. The Data Logger must respond with either an Ack PDU or a Nak PDU.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
inrtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
requester	pointer to SimulationAddress	address.h
Calls		
Function	**Where Described**	
new_rtc	Sec. 2.2.2.5.14	
logger_connect	Sec. 2.2.2.2.23	
init_pdu	Sec. 2.2.2.5.19	
init_apdu	Sec. 2.2.2.5.18	
send_rtc	Sec. 2.2.2.5.17	

Table 2.2-88: connect_request_handler Information.

2.2.2.5.22 disconnect_handler

This routine disconnects the Data Logger from the client. The client sends out an DisconnectRequest PDU when preparing to terminate its Data Logger connection. The Data Logger responds with an Ack PDU and disconnects.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
inrtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
Calls		
Function	**Where Described**	
new_rtc	Sec. 2.2.2.5.14	
logger_disconnect	Sec. 2.2.2.2.25	
init_pdu	Sec. 2.2.2.5.19	
init_apdu	Sec. 2.2.2.5.18	
send_rtc	Sec. 2.2.2.5.17	

Table 2.2-89: disconnect_handler Information.

2.2.2.5.23 command_request_handler

This routine handles runtime command requests from the client. Currently defined are:

Start
Continue
Suspend
Stop
Seek_relative
Play_speed

Parameters		
Parameter	**Type**	**Where Typedef Declared**
inrtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
inpdu	pointer to LoggerPDU	p_logger.h
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
files	file_info	Sec. 2.2.1.8 util.h
msg	pointer to char	standard
rcd	pointer to LoggerStartRTC pointer to LoggerSeekRTC pointer to LoggerSpeedRTC	p_logger.h
seconds	int	standard
dseconds	double	standard

Errors	
Error	**Reason for Error**
logger_error	Failed to start the Logger
Calls	
Function	**Where Described**
new_rtc	Sec. 2.2.2.5.14
logger_op_error	Sec. 2.2.2.2.22
init_pdu	Sec. 2.2.2.5.19
init_apdu	Sec. 2.2.2.5.18
send_rtc	Sec. 2.2.2.5.17
init_file_info	Sec. 2.2.2.8.1
parse_start_file_info	Sec. 2.2.2.5.24
logger_start	Sec. 2.2.2.2.29
logger_error	Sec. 2.2.2.5.3
logger_continue	Sec. 2.2.2.2.24
logger_suspend	Sec. 2.2.2.2.30
logger_stop	Sec. 2.2.2.2.28
logger_seek_relative	Sec. 2.2.2.2.26
logger_speed	Sec. 2.2.2.2.27

Table 2.2-90: command_request_handler Information.

2.2.2.5.24 parse_start_file_info

This routine translates logger PDU's containing start commands into file_info structures to be passed to logger_start.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
rcd	pointer to LoggerStartRTC	p_logger.h
files	pointer to file_info	Sec. 2.2.1.8 util.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
len	unsigned	standard
i	int	standard
Called By		
Function	**Where Described**	
command_request_handler	Sec. 2.2.2.5.23	

Table 2.2-91: parse_start_file_info Information.

2.2.2.5.25 status_request_handler

This routine passes back status information to the client in response to StatusRequest PDU's. The client sends out an StatusRequest PDU when requesting information about its current operating status. The Data Logger must respond with a StatusReply PDU.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
inrtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
Calls		
Function	**Where Described**	
new_rtc	Sec. 2.2.2.5.14	
do_status_reply	Sec. 2.2.2.5.19	
init_apdu	Sec. 2.2.2.5.18	
send_rtc	Sec. 2.2.2.5.17	

Table 2.2-92: status_request_handler Information.

2.2.2.5.26 do_information

This routine builds a logger PDU containing information to be passed back to the client. This information can either be about the results of some request for logger action (*i.e.*, an error message) or be some purely informational message.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pdu	pointer to LoggerPDU	p_logger.h
is_error	BOOL	standard
errmsg	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
var	pointer to LoggerInformationVariant	p_logger.h
Calls		
Function	**Where Described**	
init_pdu	Sec. 2.2.2.5.19	
Called By		
Function	**Where Described**	
logger_msg	Sec. 2.2.2.5.2	
logger_error	Sec. 2.2.2.5.3	

Table 2.2-93: do_information Information.

2.2.2.5.27 do_clock_tick

This routine builds the ClockTick PDU that the logger uses to pass the current exercise time back to its client.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pdu	pointer to LoggerPDU	p_logger.h
sec_offset	unsigned long	standard
start_dt	pointer to LoggerTime	p_logger.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
var	pointer to LoggerClockTickVariant	p_logger.h
Calls		
Function	**Where Described**	
init_pdu	Sec. 2.2.2.5.19	
Called By		
Function	**Where Described**	
send_clock_tick	Sec. 2.2.2.5.29	

Table 2.2-94: do_clock_tick Information.

2.2.2.5.28 *do_status_reply*

This routine builds the StatusReply PDU that the logger uses to reply to a client's request for status information.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
start_dt	pointer to LoggerTime	p_logger.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
activity	LoggerActivity	p_logger.h
operation	LoggerOperation	p_logger.h
state	LoggerState	p_logger.h
racal	int	standard
speed	int	standard
file_id	pointer to char	standard
project_id	pointer to char	standard
comment	pointer to char	standard
fio_head	pointer to fio_header	Sec. 2.1.4.1.2 fioheader.h
var	pointer to LoggerStatusReplyVariant	p_logger.h

Calls	
Function	**Where Described**
init_pdu	Sec. 2.2.2.5.19
logger_status	Sec. 2.2.2.2.31
Called By	
Function	**Where Described**
status_request_handler	Sec. 2.2.2.5.25

Table 2.2-95: do_status_reply Information.

2.2.2.5.29 send_clock_tick

This routine sends the clock tick to the client.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
msec_offset	unsigned long	standard
start_dt	pointer to LoggerTime	p_logger.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
rtc	pointer to fio_packet	Sec. 2.1.4.1.1 fiolib.h
apdu	pointer to AssociationPDU	p_assoc.h
pdu	pointer to LoggerPDU	p_logger.h
sec_offset	unsigned long	standard
dsec_offset	double	standard
Calls		
Function	**Where Described**	
new_rtc	Sec. 2.2.2.5.14	
do_clock_tick	Sec. 2.2.2.5.27	
init_apdu	Sec. 2.2.2.5.18	
send_rtc	Sec. 2.2.2.5.17	
Called By		
Function	**Where Described**	
logger_clock_tick	Sec. 2.2.2.2.21	

Table 2.2-96: do_status_reply Information.

2.2.2.6 timer.c

This file handles the Data Logger's timer. It provides the speed-up and freeze functions of the Data Logger during playback. Note that are three times:

(1) Real time, *i.e.*, clock time
(2) CMC time is the time on the CMC driver
(3) Exercise time is the real time elapsed from the start of the exercise to the time stamp on the current packet (it may be adjusted by a certain factor)

Note also that it is only necessary that the time since the last change in rate be used to modify the exercise time, since the CMC time will be changed every time the rate changes.

There are several interactions between the procedures:

- -- **freeze_time()** and **unfreeze_time()** affect only **time_of_freeze.**
- -- **set_fast_time_factor()** affects **time_of_fast** and **fast_factor** and is dependent upon **time_of_freeze**
- -- **set_time()** affects **time_of_fast** and **time_of_freeze** if set.
- -- **get_time()** is affected by **time_of_fast, time_of_freeze,** and **fast_factor.**

It is important to note that **on_time()** depends upon the 16 bit Network clock for timekeeping purposes. According to Network Interface Library documentation, a call to get time must be made at least once every 60 seconds for the 16-bit network clock to remain accurate.

Includes:

<stdio.h>
"sys/types.h>
"global.h"
"cmc.h"
"timer.h"

Variable Declarations:

fast_factor
time_of_fast
time_of_freeze

2.2.2.6.1 exe_to_cmc_time

This routine converts the exercise time to CMC time.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
exe_time	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ct	int	standard
Return Values		
Return Value	**Type**	**Meaning**
ct	int	The cmc time of the specified exercise time.
Called By		
Function		**Where Described**
logger_continue		Sec. 2.2.2.2.24

Table 2.2-97: exe_to_cmc_time Information.

2.2.2.6.2 cmc_to_exe_time

This routine converts the CMC time to exercise time.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cmc_time	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
et	int	standard
Return Values		
Return Value	**Type**	**Meaning**
et	int	The exercise time of the specified cmc time.
Called By		
Function		**Where Described**
get_time		Sec. 2.2.2.6.7
freeze_time		Sec. 2.2.2.6.8
unfreeze_time		Sec. 2.2.2.6.9
on_time		Sec. 2.2.2.6.10

Table 2.2-98: cmc_to_exe_time Information.

2.2.2.6.3 set_fast_time_factor

This routine sets the CMC time to exercise time conversion factor.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
factor	int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ect	int	standard
Return Values		
Return Value	**Type**	**Meaning**
ect	int	
Calls		
Function		**Where Described**
get_time		Sec. 2.2.2.6.7
cmc_settime		Sec. 2.2.2.1.11
Called By		
Function		**Where Described**
logger_speed		Sec. 2.2.2.2.27

Table 2.2-99: set_fast_time_factor Information.

2.2.2.6.4 get_fast_time_factor

This routine gets the CMC time to exercise time conversion factor.

Called By	
Function	**Where Described**
logger_status	Sec. 2.2.2.2.31
logger_playback_state	Sec. 2.2.2.2.18

Table 2.2-100: get_fast_time_factor Information.

2.2.2.6.5 reset_time

This routine resets the CMC time to 0, the fast_factor to 1, the time_of_fast to 0, and the time_of_freeze to 0.

Calls	
Function	**Where Described**
cmc_settime	Sec. 2.2.2.1.11
Called By	
Function	**Where Described**
logger_start	Sec. 2.2.2.2.29

Table 2.2-101: reset_time Information.

2.2.2.6.6 set_time

This routine changes time to begin playing at the passed Exercise time.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
exe_time	int	standard
Return Values		
Return Value	**Type**	**Meaning**
exe_time	int	the Exercise time; procedure was successful
Calls		
Function	**Where Described**	
cmc_settime	Sec. 2.2.2.1.11	
Called By		
Function	**Where Described**	
logger_seek_relative	Sec. 2.2.2.2.26	
logger_continue	Sec. 2.2.2.2.24	

Table 2.2-102: set_time Information.

2.2.2.6.7 get_time

This routine gets the current Exercise time.

Return Values		
Return Value	**Type**	**Meaning**
time_of_freeze	int	the time of freeze
cmc_to_exe_time (cmc_gettime32)	int	the current Exercise time

Calls	
Function	**Where Described**
cmc_to_exe_time	Sec. 2.2.2.6.2
cmc_gettime32	Sec. 2.2.2.1.10

Called By	
Function	**Where Described**
logger_suspend	Sec. 2.2.2.2.30
logger_status	Sec. 2.2.2.2.31
ccatcher	Sec. 2.2.2.2.34
logger_stop	Sec. 2.2.2.2.28
get_time	Sec. 2.2.2.2.7
set_fast_time_factor	Sec. 2.2.2.6.3

Table 2.2-103: get_time Information.

2.2.2.6.8 freeze_time

This routine freezes the Exercise time.

Return Values		
Return Value	**Type**	**Meaning**
time_of_freeze	int	the Exercise time at the time of freeze

Calls	
Function	**Where Described**
cmc_to_exe_time	Sec. 2.2.2.6.2
cmc_gettime32	Sec. 2.2.2.1.10

Called By	
Function	**Where Described**
logger_suspend	Sec. 2.2.2.2.30

Table 2.2-104: freeze_time Information.

2.2.2.6.9 unfreeze_time

This routine unfreezes the Exercise time.

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
et	int	standard
Return Values		
Return Value	**Type**	**Meaning**
et	int	the Exercise time at the time of unfreezing
Calls		
Function		**Where Described**
cmc_settime		Sec. 2.2.2.1.11
exe_to_cmc_time		Sec. 2.2.2.6.1
Called By		
Function		**Where Described**
logger_continue		Sec. 2.2.2.2.24

Table 2.2-105: unfreeze_time Information.

2.2.2.6.10 on_time

This routine takes an Exercise time, converts it to CMC time, and returns TRUE if the resulting time has not passed.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
time	pointer to int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ct	int	standard
Return Values		
Return Value	**Type**	**Meaning**
time>ct	BOOL	if TRUE, the specified time has not passed; if FALSE, the specified time has passed
Calls		
Function		**Where Described**
exe_to_cmc_time		Sec. 2.2.2.6.1
cmc_gettime16		Sec. 2.2.2.1.9

Called By	
Function	Where Described
logger_playback_state	Sec. 2.2.2.2.18

Table 2.2-106: on_time Information.

2.2.2.6.11 time_string

This routine converts an integer time into a string.

Parameters		
Parameter	Type	Where Typedef Declared
msec	int	standard
Internal Variables		
Variable	Type	Where Typedef Declared
time_buf[64]	char	standard
Return Values		
Return Value	Type	Meaning
time_buf	pointer to char	the time string
Called By		
Function	Where Described	
logger_suspend	Sec. 2.2.2.2.30	
logger_speed	Sec. 2.2.2.2.27	

Table 2.2-107: time_string Information.

2.2.2.7 user.c

This file provides the user interface and help functions for the stand-alone Data Logger.

Includes:

<stdio.h>
<ctype.h>
<sys/ioctl>
<sys/types.h>
<sys/stat.h>
"fiolib.h"
"p_logger.h"
"global.h"
"util.h"
"user.h"

Imported Procedure Declarations:

fopen()
free()
malloc()
perror()
strncpy()
strrchr()

strtok()

Variable Declarations:
- errno
- sys_errlist[]
- msgbuf[MAX_LINE_LEN+1]
- helpbuf[MAX_LINE_LEN+1]
- input[MAX_LINE_LEN+1]
- help_stream

Static Procedure Declarations:
- **collect_diskfile_info()**
- **collect_tapefile_info()**
- **user_putstr()**
- **prompt_user()**
- **help_function()**
- **can_access_file()**
- **vlaunch()**

Defines:

Formatting Strings:
- INDENT_STRING
- PROMPT_INDENT
- REPLY_INDENT
- HELP_INDENT

Help Strings:
- HELP_RACAL_STARTTIME_INFO
- HELP_CONTIGUOUS_DISKINFO
- HELP_FILENAME_DISKINFO
- HELP_FILESIZE_DISKINFO
- HELP_LOOPING_DISKINFO
- HELP_CONTINUOUS_TAPEINFO
- HELP_DRIVES_TAPEINFO
- HELP_SIZE_TAPEINFO
- HELP_MEDIUM_FILEINFO
- HELP_CONTINUE_OR_QUIT
- HELP_EXERCISEID_INFO
- HELP_FIOHEADER_INFO
- HELP_NUMEXERCISES_INFO
- HELP_PROTOCOL_TYPE_INFO
- HELP_YES_OR_NO
- HELP_STRING_CHAR
- VALID_EXAMPLE_CHAR
- HELP_FILE_NAME

Tape Strings:
- R625SMT0
- R625SMT1
- MAX_LINE_LEN

2.2.2.7.1 collect_racal_time_info

This collects information from the user about the racal recorder.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
racal_time	pointer to int	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
done	BOOL	standard
Calls		
Function	**Where Described**	
user_putstr	Sec. 2.2.2.7.10	
prompt_user	Sec. 2.2.2.7.11	
string_to_seconds	Sec. 2.1.3.2.3	
Called By		
Function	**Where Described**	
logger_start	Sec. 2.2.2.2.29	

Table 2.2-108: collect_racal_time_info Information.

2.2.2.7.2 collect_fiofile_info

This routine collects information from the user about the type of fiofile to open for reading or writing.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
files	pointer to file_info	Sec. 2.2.1.8 util.h
dfltnm	pointer to char	standard
write	BOOL	standard
debug	BOOL	standard
query_all	BOOL	standard
Calls		
Function	**Where Described**	
prompt_user	Sec. 2.2.2.7.11	
collect_diskfile_info	Sec. 2.2.2.7.3	
collect_tapefile_info	Sec. 2.2.2.7.4	
Called By		
Function	**Where Described**	
init_logger_for_playback	Sec. 2.2.2.2.8	
init_logger_for_recording	Sec. 2.2.2.2.9	
init_logger_for_copying	Sec. 2.2.2 2.7	

Table 2.2-109: collect_fiofile_info Information.

2.2.2.7.3 collect_diskfile_info

This routine collects information from the user about the specified disk file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
files	pointer to file_info	Sec. 2.2.1.8 util.h
dfltnm	pointer to char	standard
write	BOOL	standard
debug	BOOL	standard
query_all	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
done	BOOL	standard
i	int	standard
nvolumes	int	standard
filesz	int	standard
filenm	pointer to char	standard
s	pointer to char	standard
Errors		
Error	**Reason for Error**	
Logger WARNING:	Specified file already exists	
Calls		
Function	**Where Described**	
prompt_user	Sec. 2.2.2.7.11	
user_putstr	Sec. 2.2.2.7.10	
can_access_file	Sec. 2.2.2.7.14	
yes_or_no	Sec. 2.2.2.7.9	
Called By		
Function	**Where Described**	
collect_fiofile_info	Sec. 2.2.2.7.2	

Table 2.2-110: collect_diskfile_info Information.

2.2.2.7.4 collect_tapefile_info

This routine collects information from the user about the specified tape file.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
files	pointer to file_info	Sec. 2.2.1.8 util.h
query_all	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
tape_drives	int	standard
tape_size	int	standard

Calls	
Function	**Where Described**
prompt_user	Sec. 2.2.2.7.11
user_putstr	Sec. 2.2.2.7.10
Called By	
Function	**Where Described**
collect_fiofile_info	Sec. 2.2.2.7.2

Table 2.2-111: collect_tapefile_info Information.

2.2.2.7.5 collect_cmcsubscribe_info

This routine collects information about the specified exercises from the user.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
sim_protocol	pointer to BOOL	standard
voice_protocol	pointer to BOOL	standard
num_exercises	pointer to short	standard
exercises	pointer to char	standard
num_stdprotocols	pointer to short	standard
num_nstdprotocols	pointer to short	standard
debug	BOOL	standard
query_all	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
str[32]	char	standard
id	int	standard
i	int	standard
Calls		
Function	**Where Described**	
prompt_user	Sec. 2.2.2.7.11	
user_putstr	Sec. 2.2.2.7.10	
Called By		
Function	**Where Described**	
init_logger_for_recording	Sec. 2.2.2.2.9	

Table 2.2-112: collect_cmcsubscribe_info Information.

2.2.2.7.6 collect_fioheader_info

This routine collects information from the user for the DataProbe header in the fio_header.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
file_id	pointer to char	standard
fileid_sz	int	standard
comment	pointer to char	standard
comment_sz	int	standard
project	pointer to char	standard
project_sz	int	standard

Calls	
Function	**Where Described**
prompt_user	Sec. 2.2.2.7.11

Called By	
Function	**Where Described**
init_logger_for_recording	Sec. 2.2.2.2.9

Table 2.2-113: collect_fioheader_info Information.

2.2.2.7.7 collect_playback_info

This routine collects information from the user about whether to play back the stealth protocol.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
play_stealth	pointer to BOOL	standard
query_all	BOOL	standard

Calls	
Function	**Where Described**
prompt_user	Sec. 2.2.2.7.11
user_putstr	Sec. 2.2.2.7.10

Called By	
Function	**Where Described**
init_logger_for_playback	Sec. 2.2.2.2 .8

Table 2.2-114: collect_playback_info Information.

2.2.2.7.8 continue_or_quit

This routine asks the user whether they want to continue or quit.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
str	pointer to char	standard
help_str	pointer to char	standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	BOOL	standard
Return Values		
Return Value	**Type**	**Meaning**
retval	BOOL	If retval = TRUE, user wants to continue; If retval = FALSE, user wants to quit
Calls		
Function	**Where Described**	
prompt_user	Sec. 2.2.2.7.11	
Called By		
Function	**Where Described**	
logger_copying_state	Sec. 2.2.2.2.16	

Table 2.2-115: continue_or_quit Information.

2.2.2.7.9 yes_or_no

This routine asks the user "yes or no".

Parameters		
Parameter	**Type**	**Where Typedef Declared**
str	pointer to char	standard
help_str	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
retval	BOOL	standard
Return Values		
Return Value	**Type**	**Meaning**
retval	BOOL	If retval = TRUE, user responded with YES; If retval = FALSE, user responded with NO
Calls		
Function	**Where Described**	
prompt_user	Sec. 2.2.2.7.11	

Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1
init_logger_for_playback	Sec. 2.2.2.2.8
init_logger_for_recording	Sec. 2.2.2.2.9
init_logger_for_copying	Sec. 2.2.2.2.7
collect_diskfile_info	Sec. 2.2.2.7.3
help_function	Sec. 2.2.2.7.12

Table 2.2-116: yes_or_no Information.

2.2.2.7.10 user_putstr

This routine formats output.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
str	pointer to char	standard
indent_level	int	standard
newline	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	standard
Called By		
Function	**Where Described**	
collect_racal_time_info	Sec. 2.2.2.7.1	
collect_tapefile_info	Sec. 2.2.2.7.4	
collect_cmcsubscribe_info	Sec. 2.2.2.7.5	
collect_playback_info	Sec. 2.2.2.7.7	
prompt_user	Sec. 2.2.2.7.11	
help_function	Sec. 2.2.2.7.12	
collect_diskfile_info	Sec. 2.2.2.7.3	

Table 2.2-117: user_putstr Information.

2.2.2.7.11 prompt_user

This routine prompts the user.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
prompt	pointer to char	standard
user_response	pointer to char	standard
help_str	pointer to char	standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
done	BOOL	standard
buffer[MAX_LINE_LEN+1]	char	standard
s	pointer to char	standard

Calls	
Function	**Where Described**
user_putstr	Sec. 2.2.2.7.10
help_function	Sec. 2.2.2.7.12

Called By	
Function	**Where Described**
collect_racal_time_info	Sec. 2.2.2.7.1
collect_fiofile_info	Sec. 2.2.2.7.2
collect_diskfile_info	Sec. 2.2.2.7.3
collect_tapefile_info	Sec. 2.2.2.7.4
collect_cmcsubscribe_info	Sec. 2.2.2.7.5
collect_playback_info	Sec. 2.2.2.7.7
continue_or_quit	Sec. 2.2.2.7.8
yes_or_no	Sec. 2.2.2.7.9
collect_fioheader_info	Sec. 2.2.2.7.6

Table 2.2-118: prompt_user Information.

2.2.2.7.12 help_function

This routine handles the on-line help function.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
help_str	pointer to cnar	standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
done	BOOL	standard
iof	BOOL	standard
s	pointer to char	standard

Calls	
Function	**Where Described**
user_putstr	Sec. 2.2.2.7.10
yes_or_no	Sec. 2.2.2.7.9

Called By	
Function	**Where Described**
prompt_user	Sec. 2.2.2.7.11

Table 2.2-119: help_function Information.

2.2.2.7.13 init_user

This routine initializes the user.

Called By	
Function	**Where Described**
main	Sec. 2.2.2.2.1

Table 2.2-120: init_user Information.

2.2.2.7.14 can_access_file

This routine checks the path name of the file to find out if the specified file exists and returns TRUE if if it does.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
path	pointer to char	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
stat_buf	struct stat	standard
Return Values		
Return Value	**Type**	**Meaning**
msgbuf	pointer to char	the specified file exists
NULL	pointer to char	the specified file does not exist
Errors		
Error	**Reason for Error**	
Logger WARNING	- Can't access specified file - Specified file is a directory	
Called By		
Function	**Where Described**	
collect_diskfile_info	Sec. 2.2.2.7.3	

Table 2.2-121: can_access_file Information.

2.2.2.7.15 vlaunch

This routine is a method of executing a command to the UNIX shell; it is a fast version of launch. The first item in the command must be the complete pathname of a program. *ccmd[128]* is the mugified copy of cmd. *prog* is the file to be EXECed. *argv[128]* is the arguments to prog.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
cmd	pointer to char	standard

Internal Variables		
Variable	**Type**	**Where Typedef Declared**
ccmd[128]	char	standard
prog	pointer to char	standard
argv[128]	pointer to char	standard
count	int	standard
Return Values		
Return Value	**Type**	**Meaning**
0	int	launch was successful
-1	int	launch failed

Table 2.2-122: vlaunch Information.

2.2.2.8 util.c

This file contains low level utilities used by the Data Logger.
Includes:

<stdio.h>
"global.h"
"util.h"

Imported procedure declarations:

free()

2.2.2.8.1 init_file_info

This routine sets the following file characteristics:

nvolumes = 0
curvol = 1
contig = FALSE
covert = 0
looping = FALSE

Parameters		
Parameter	**Type**	**Where Typedef Declared**
files	pointer to file_info	Sec. 2.2.1.8 util.h
free_mem	BOOL	standard
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
i	int	standard
Called By		
Function	**Where Described**	
init_logger_for_playback	Sec. 2.2.2.2.8	
init_logger_for_recording	Sec. 2.2.2.2.9	
init_logger_variables	Sec. 2.2.2.2.4	
command_request_handler	Sec. 2.2.2.5.23	

Table 2.2-123: init_file_info Information.

2.2.2.9 logfilt.c

This file is used to direct the CMC driver card. It contains the filters that are downloaded to the CMC card while the logger is running. The following PDU's are allowed to pass through the filter:

1) All packets belonging to the 'protocolLogger' protocol.
2) All packets belonging to the 'protocolSim' protocol if the logger has explicitely asked for their MulticastGroupID (exercise ID).
3) All packets belonging to the current version of other standard or proprietary protocols if the logger has explicitely asked for that protocol.

Includes:

"p_All.h"
"enpif.h"
"network.h"
"netfilter.h"
"cmc.h"

2.2.2.9.1 do_init

This routine is called by the CMC card to initialize the link with the logger.

2.2.2.9.2 do_packet_from_host

This routine filters a packet from the host.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pkt	pointer to short	standard
Return Values		
Return Value	**Type**	**Meaning**
SEND_PACKET	int	PDU was accepted

Table 2.2-124: do_packet_from_host Information.

2.2.2.9.3 do_packet_from_network

This routine filters a packet from the network.

Parameters		
Parameter	**Type**	**Where Typedef Declared**
pkt	pointer to NetworkPacket	network.h
Internal Variables		
Variable	**Type**	**Where Typedef Declared**
apdu	pointer to register AssociationPDU	p_assoc.h
protocol	register int	standard
exercise	register int	standard
retval	int	standard

Return Values		
Return Value	**Type**	**Meaning**
retval	int	if retval = SEND_PACKET, the PDU was accepted; if retval = FREE_PACKET, the PDU was not accepted

Calls	
Function	**Where Described**
GET_DATA_PTR	network.h
ACCEPT_EXERCISE	Sec. 2.2.1.1 cmc.h (macro definition)
ACCEPT_PROTOCOL	Sec. 2.2.1.1 cmc.h (macro definition)

Table 2.2-125: do_packet_from_network Information.

APPENDIX A: FIO DATA FILE FORMAT

The data file is composed of a series of 24k blocks, as illustrated in Figure A-1 The first 24k block is called the `fio_file_header` and is defined in "fioheader.h". The next 24k blocks are the `fio_records`, also called the Data Blocks. A variable number of these 24k blocks are each formatted as shown in Figure A-2. Each record is comprised of a 4 byte header (consisting of a length field) and a variable number of fio packets, each of arbitrary length. The `fio_record` is declared in "fiofile.h".

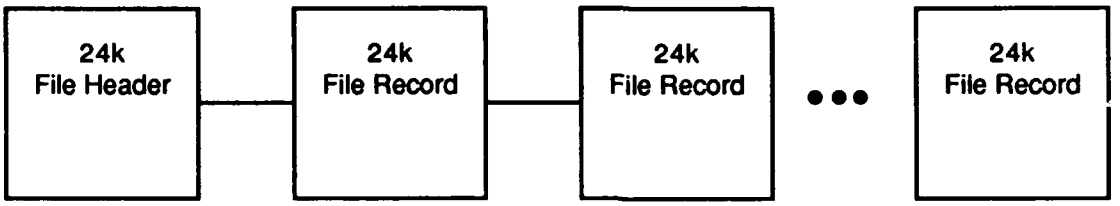


Figure A-1: SIMNET Data File Structure

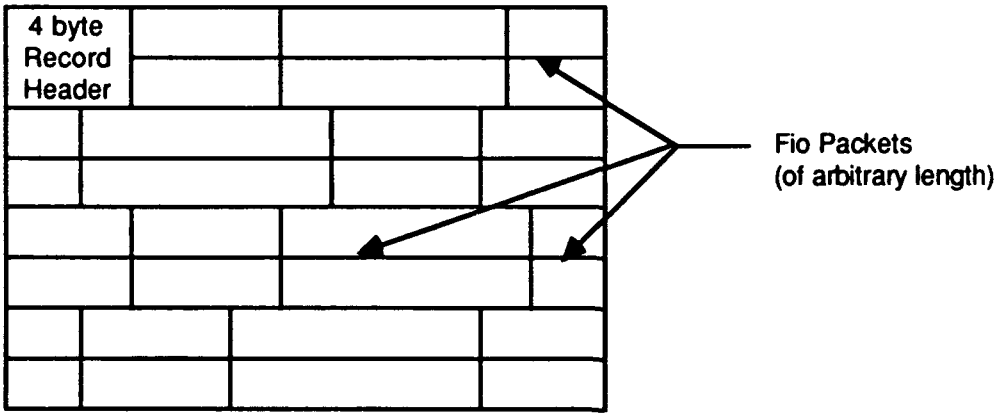


Figure A-2: File Record Structure

The `fio_header` includes the following fields:

```
fio_header
     dp_header                    -- Data Probe header consists of:
          tape_type                  -- ADP, SIM, etc.
          run_number
          project_id                 -- SIMNET, etc.
          comment                    -- String from user
          date_of_run                -- When created
          tape_number                -- Volume number of tape
          bits_per_word
          frame_size
          header_size
          date_of_copy
          time_of_run
          time_of_launch

     fiolib_version_chksum      -- For checking compatability
     major_rev                  -- DataLogger version # used to record
     minor_rev
     user_info_offset           -- The offset to the start of user defined area
     user_info_bytes            -- Number of bytes in user defined area
     ui_heap                    -- 3rd to last block: user defined info
          file_id                    -- The file id number
          nexercise_ids              -- The number of exercises
          exercise_ids               -- Which exercises to record
          command_str                -- Char string typed to Unix prompt
```

The total number of remaining 24k blocks in the series varies from file to file and is defined in the variable `FIO_RECORD_BLOCKS`. Each of these data blocks is called an `fio_record`. The `fio_record` consists of enough variable length packets to fill the 24k data block. The `fio_record` is defined in "fiofile.h" and includes the following fields:

```
fio_record
     record_header              -- 4 byte length field

-- The remainder of the record is filled with Packets of the following format:

          fio_packet_header     -- Packet header consists of:
               time                  -- time stamp for play back
               num_bytes             -- # of bytes in data portion
               from                  -- from NetworkAddress
               to                    -- to NetworkAddress
               type                  -- ethernet packet protocol type
          data                  -- Packet data
```

APPENDIX B: PVD DATA LOGGER COMMUNICATIONS PROTOCOL

This protocol is intended to handle the communication needs of the PVD and the Data Logger when the Data Logger is being controlled by from the PVD in 'server' mode. Logger/PVD PDUs share the CMC network with the standard simulation protocol PDUs. However, Logger/PVD communications take place on a unique multi-caste address consisting of the following:

```
MulticastGroupID          0
AssociationUserProtocol   loggerProtocolNumber
```

The three phases of Logger/PVD communications are:

1) Connect Phase
2) Command Phase
3) Disconnect Phase

During the first phase, the Logger will respond to all PDU's arriving on its Multi-cast group. Thereafter it responds only to PDU's which are explicitely directed to it.

In the discussion of the Logger/PVD protocol outlined below, it is assumed that the Logger and PVD will be using the services of the Association Protocol for their communications. All PDU's are assumed to be sent via the POINT_TO_POINT_DATAGRAM (ptp datagram) service unless otherwise stated.

```
constant   loggerAcceptAll          256
constant   loggerMaxCommentSz       24
constant   loggerMaxErrorStringSz   128
constant   loggerMaxExerciseIDs     8
constant   loggerMaxFileNameSz      64
constant   loggerMaxFileVolumes     2
constant   loggerMaxHostNameSz      32
constant   loggerMaxProjIDSz        7
constant   loggerMulticastGroupID   0
```

PDUKIND `loggerAvailRequestPDUKind`
OVERVIEW
The `loggerAvailRequestPDUKind` is sent by the PVD while preparing to connect to a Logger. It is sent on the Logger/PVD multi-cast address using the broadcast Datagram (as opposed to the ptpDatagram) service.

All Loggers which are operating in server mode and which are not already connected to a PVD must respond with a `'loggerAvailReplyPDUKind'` PDU. Both the Request and the Reply are sent using the ptpDatagram service. The PVD must wait a reasonable amount of time for the Loggers to respond before assuming no other Loggers are out there.

PDUKIND `loggerConnectRequestPDUKind`
OVERVIEW
The PVD sends a `'loggerConnectPDUKind'` to the particular Logger to which it wants to connect using the ptpDatagram service of the Association Protocol.

The Logger responds with either a 'loggerAckPDUKind' or a 'loggerNakPDUKind' using the ptpDatagram service.

PDUKIND loggerCommandRequestPDUKind
OVERVIEW
The PVD sends the loggerCommandRequestPDUKind to the Logger to which it is connected in order to specify runtime commands. The PVD uses the ptpDatagram service of the Association Protocol. The following runtime commands are currently defined in the PVD/Logger protocol:

```
START
CONTINUE
SUSPEND
STOP
SEEK_RELATIVE
PLAY_SPEED
```

PDUKIND loggerStatusRequestPDUKind
OVERVIEW
The PVD sends a 'loggerStatusRequestPDUKind' to the Logger to which it is connected to gather information about its current operating status. The PVD uses the ptpDatagram service of the Association Protocol.

The Logger responds with a 'loggerStatusReplyPDUKind' using the ptpDatagram service.

PDUKIND loggerDisconnectRequestPDUKind
OVERVIEW
The PVD sends a 'loggerDisconnectPDUKind' to the Logger to which it is connected to terminate its connection. The PVD uses the ptpDatagram service of the Association Protocol.

The Logger responds with a 'loggerAckPDUKind' using the ptpDatagram service.

PDUKIND loggerAckPDUKind
OVERVIEW
The Logger uses this PDU to reply to successful PVD requests.

PDUKIND loggerNakPDUKind
OVERVIEW
The Logger uses this PDU to reply to failed PVD requests.

PDUKIND loggerAvailReplyPDUKind
OVERVIEW
The Logger uses this PDU to reply to PVD Availability Requests.

PDUKIND loggerStatusReplyPDUKind
OVERVIEW
The Logger uses this PDU to reply to PVD Status Requests.

PDUKIND loggerInformationPDUKind

OVERVIEW
The Logger uses this PDU to communicate unsolicited information to the PVD using the ptpDatagram service.

PDUKIND `loggerClockTickPDUKind`
OVERVIEW
The Logger uses this PDU to communicate timing information to the PVD using the ptpDatagram service.

Kinds of loggerProtocol PDUs:

-- loggerProtocol PDUs with which the PVD sends requests to the Logger.

```
type LoggerPDUKind enum (8) {
     loggerAvailRequestPDUKind   (1),
     loggerConnectRequestPDUKind,
     loggerCommandRequestPDUKind,
     loggerStatusRequestPDUKind,
     loggerDisconnectRequestPDUKind,
```

-- loggerProtocol PDUs with which the Logger replies to PVD requests:

```
     loggerAckPDUKind,
     loggerNakPDUKind,
     loggerAvailReplyPDUKind,
     loggerStatusReplyPDUKind,
```

-- loggerProtocol PDUs with which the Logger sends unsolicited information to the PVD:

```
     loggerInformationPDUKind,
     loggerClockTickPDUKind,
     loggerLastPDUKind
 }
```

The following defines the set of possible file mediums for Logger files:

```
type LoggerMedium enum (3) {
     diskMedium (1),
     nineTrackTapeMedium,
     cassetteTapeMedium
}

 type LoggerTime sequence {
     year      UnsignedInteger  (8), -- Offset from 1900
     month     UnsignedInteger  (8),
     day       UnsignedInteger  (8),
     hour      UnsignedInteger  (8),
     minute    UnsignedInteger  (8),
     second    UnsignedInteger  (8),
     tzone     UnsignedInteger  (8), -- Hrs West of GMT (-1 if unknown)
```

```
      dstime    UnsignedInteger  (8)    -- True if daylight savings time.
}

type LoggerStartRTC sequence {

      starttime     LoggerTime,          --Recording Only: Base time of an
                                         exercise.

      contig        Boolean,             -- Recording to Disk Only: Use contiguous
                                         disk files.

      looping       Boolean,             -- Recording to Disk: When the end of
                                         file is reached, return to the start and
                                         continue recording.
                                         --Play back from Disk: Follow the loop
                                         pointer to the logical start of file. When
                                         the end of file is reached, go to the
                                         beginning of the file and continue playing
                                         back until the loop pointer is reached. (i.e.
                                         the loop pointer marks the logical
                                         beginning and end of file.
                                         --Record and Play Back to Tape:
                                         Swapping between available tape drives.

      recording     Boolean,             -- True => Recording;
                                         -- False => Play back.

      promiscuous   Boolean,
      racal         Boolean,
      medium        LoggerMedium,
      nvolumes UnsignedInteger(8),  -- Count of the 'flnm' array entries used.

      num_exercises Integer(16),   -- Recording: exercise ids is an array of
                                         1-byte integers in the range 1 .. 255. A
                                         value of 'loggerAcceptAll' indicates that all
                                         exercise ids are to be accepted.

      exercises  array(loggerMaxExerciseIDs) of Character (8),

      num_stdprotocols Integer (16),
                                         --Recording: A value of 'loggerAcceptAll'
                                         indicates that all standard or non-standard
                                         protocols are to be accepted. A value of 0
                                         indicates that none are to be accepted. Any
                                         value other than 'loggerAcceptAll' will be
                                         interpreted as 0.

      num_nstdprotocols Integer (16),
                                         -- Recording to Contiguous Disk files:
                                         size in bytes. Recording to Tape:
                                         1=>600 ft 2=>1200 ft; 3=>2400 ft reels

      size array(loggerMaxFileVolumes) of UnsignedInteger (32),
```

```
        flnm array(loggerMaxFileVolumes,loggerMaxFileNameSz) of
        Character (8),
                                        -- Disk: Names of separate Unix files
                                        concatenated to create the logical SIMNET
                                        data file. Max of 'loggerMaxFileVolumes'
                                        files can be involved.
                                        -- Tape: Names of tape drives being used.
                                        A max of 'loggerMaxFileVolumes' drives
                                        can be involved.

        projID array(loggerMaxProjIDSz) of Character (8),
                                        -- Header Information

        comment         array(loggerMaxCommentSz) of Character (8),

        rcdVoiceProtocolType  Boolean,
                                        -- Used when the logger is being initialized
                                        for Recording. Controls whether packets
                                        belonging to the Voice Protocol Type are
                                        recorded.


        rcdSimProtocolType    Boolean,
                                        --Used when the logger is being initialized
                                        for Recording. Controls whether packets
                                        belonging to the Simulation Protocol Type
                                        are recorded.

        playStealthProtocol   Boolean,
                                        --Used when the logger is being initialized
                                        for Play Back. Controls whether packets
                                        belonging to the Stealth Protocol of the
                                        Simulation Protocol Type are recorded.

        unused (5)
}

type LoggerSeekRTC sequence {
        relative_curloc         Integer (32),
        seconds                 Integer (32)
}

type LoggerSpeedRTC sequence {
        factor                  Integer (32)
}
```

The following defines the set of possible runtime commands which the PVD can send to the Data Logger.

```
type LoggerRTCType enum (8) {
        startRTCType (1),
```

```
      continueRTCType,
      suspendRTCType,
      stopRTCType,
      seekRelativeRTCType,
      playSpeedRTCType
}

type LoggerCommandVariant sequence {
      command    LoggerRTCType,
      unused     (24),

      info choice (command) of {

            when (startRTCType)
                  startInfo LoggerStartRTC,
            when (seekRelativeRTCType)
                  seekInfo  LoggerSeekRTC,
            when (playSpeedRTCType)
                  playInfo  LoggerSpeedRTC
      }
}


type LoggerActivity enum (2) {
      loggerNoActivity (1),
      loggerRecording,
      loggerPlayback,
      loggerCopying
}

type LoggerOperation enum (2) {
      loggerInteractive (1),
      loggerServer
}

type LoggerState enum (2) {
      loggerDisconnected (1),
      loggerConnected,
      loggerSuspended,
      loggerActive
}

type LoggerAvailReplyVariant sequence {
      available Boolean,
      unused    (7),
      loggerID  array(loggerMaxHostNameSz) of Character (8)
}

type LoggerStatusReplyVariant sequence {
      activity  LoggerActivity,
      oper                                LoggerOperation,
      state     LoggerState,
      racal     Boolean,
      unused    (1),
```

```
     speed      UnsignedInteger (8),        -- Time factor
     unused     (16),
     start      LoggerTime,                 -- Start time of the exercise

     offset     UnsignedInteger (32),       -- Time offset (in msec) from
                                               beginning

     packets      UnsignedInteger (32),     -- Number of packets handled
                                               since start

     file_id    array(loggerMaxFileNameSz) of Character (8),
     project_id  array(loggerMaxProjIDSz) of Character (8),
     comment    array(loggerMaxCommentSz) of Character (8),
     unused     (8)
}

type LoggerInformationVariant sequence {
     is_error   Boolean,
     unused     (7),
     error_string  array(loggerMaxErrorStringSz) of Character
                     (8)
}

type LoggerClockTickVariant sequence {
     starttime LoggerTime,
     offset    UnsignedInteger (32)
}

type LoggerProtocolVersion enum (8) {
     loggerProtocolVersionAug89 (1),
     loggerProtocolVersionJan90 (2)
}

constant  loggerProtocolVersionCurrent 2


type LoggerPDU sequence {
     version          LoggerProtocolVersion,
     kind             LoggerPDUKind,
     unused           (16),
     destination      SimulationAddress,

     variant    choice (kind) of {

          when (loggerCommandRequestPDUKind)
               commandReq      LoggerCommandVariant,

          when (loggerStatusReplyPDUKind)
               availReply      LoggerAvailReplyVariant,

          when (loggerStatusReplyPDUKind)
               statusReply     LoggerStatusReplyVariant,
```

```
            when (loggerInformationPDUKind)
                 information     LoggerInformationVariant,

            when (loggerClockTickPDUKind)
                 clockTick       LoggerClockTickVariant
      }
}
```

INDEX BY SECTION NUMBER

avail_request_handler	2.2.2.5.20
backup_past_target	2.1.4.2.14.34
bells	2.2.2.2.35
bytes_to_trcds	2.1.4.2.14.14
can_access_file	2.2.2.7.14
ccatcher	2.2.2.2.34
cmc.c	2.2.2.1
cmc.h	2.2.1.1
cmc_accept	2.2.2.1.3
cmc_filter_init	2.2.2.1.1
cmc_gettime16	2.2.2.1.9
cmc_gettime32	2.2.2.1.10
cmc_init	2.2.2.1.5
cmc_protocol_types	2.2.2.1.2
cmc_reject	2.2.2.1.4
cmc_report_status	2.2.2.1.8
cmc_settime	2.2.2.1.11
cmc_term	2.2.2.1.6
cmc_to_exe_time	2.2.2.6.2
cmc_zero_status	2.2.2.1.7
collect_cmcsubscribe_info	2.2.2.7.5
collect_diskfile_info	2.2.2.7.3
collect_fiofile_info	2.2.2.7.2
collect_fioheader_info	2.2.2.7.6
collect_playback_info	2.2.2.7.7
collect_racal_time_info	2.2.2.7.1
collect_tapefile_info	2.2.2.7.4
command_request_handler	2.2.2.5.23
concat_tbqs	2.1.4.2.14.10
connect_request_handler	2.2.2.5.21
construct_dphdr	2.1.4.2.3.5
continue_or_quit	2.2.2.7.8
create_dphdr	2.1.4.2.3.4
cvt_fio_packet	2.1.4.2.6.1
Data Logger Files	2.2
Data Logger Files	2.2.2
Data Logger Header Files	2.2.1
datetime_to_strings	2.1.3.2.5
dequeue	2.2.2.3.7
dequeue_tb	2.1.4.2.14.9
disconnect_handler	2.2.2.5.22

disk_close	2.1.4.2.10.21
disk_concat	2.1.4.2.10.6
disk_getrawfd	2.1.4.2.10.20
disk_open	2.1.4.2.10.5
disk_read	2.1.4.2.10.7
disk_seek	2.1.4.2.10.15
disk_seek_time	2.1.4.2.10.16
disk_write	2.1.4.2.10.11
do_clock_tick	2.2.2.5.27
do_eotv_action	2.1.4.2.14.2
do_information	2.2.2.5.26
do_init	2.2.2.9.1
do_packet_from_host	2.2.2.9.2
do_packet_from_network	2.2.2.9.3
do_status_reply	2.2.2.5.28
enqueue	2.2.2.3.6
enqueue_tb	2.1.4.2.14.8
eotv_phase_action	2.1.4.2.14.3
exe_to_cmc_time	2.2.2.6.1
Fast IO Library: libfio	2.1.4
find_file_slot	2.1.4.2.2.2
find_info	2.1.4.2.3.8
fiocvt.c	2.1.4.2.6
fiocvt.h	2.1.4.2.5
fiodisk.c	2.1.4.2.10
fiodisk.h	2.1.4.2.9
fioerror.h	2.1.4.1.3
fiofile.h	2.1.4.2.1
fiofile_records	2.1.4.2.10.3
fioheader.c	2.1.4.2.3
fioheader.h	2.1.4.1.2
fiolib.c	2.1.4.2.2
fiolib.h	2.1.4.1.1
fionet.c	2.1.4.2.12
fionet.h	2.1.4.2.11
fiotape.c	2.1.4.2.14
fiotape.h	2.1.4.2.14
fioutil.c	2.1.4.2.8
fioutil.h	2.1.4.2.7
fio_alloc	2.1.4.2.8.1
fio_close	2.1.4.2.2.12
fio_concat	2.1.4.2.2.6
fio_convert_header	2.1.4.2.3.2

fio_delete_info	2.1.4.2.3.9
fio_error	2.1.4.2.4.1
fio_error_string	2.1.4.2.4.2
fio_file_header	2.1.4.2.3.3
fio_file_type	2.1.4.2.2.4
fio_flush	2.1.4.2.2.11
fio_free	2.1.4.2.8.2
fio_getrawfd	2.1.4.2.2.10
fio_insert_info	2.1.4.2.3.6
fio_is_null_header	2.1.4.2.3.1
fio_open	2.1.4.2.2.5
fio_packet_size	2.1.4.2.6.2
fio_print_fiohdr	2.1.4.2.3.22
fio_read	2.1.4.2.2.7
fio_record_endtime	2.1.4.2.10.19
fio_retrieve_info	2.1.4.2.3.10
fio_seek	2.1.4.2.2.9
fio_write	2.1.4.2.2.8
fnet_close	2.1.4.2.12.6
fnet_flush	2.1.4.2.12.5
fnet_open	2.1.4.2.12.2
fnet_read	2.1.4.2.12.3
fnet_write	2.1.4.2.12.4
freeze_time	2.2.2.6.8
free_file_slot	2.1.4.2.2.3
free_queue_elt	2.2.2.3.3
free_rtc	2.2.2.5.15
get_command_fiohdr	2.1.4.2.3.14
get_datetime	2.1.3.2.6
get_fast_time_factor	2.2.2.6.4
get_fileid_fiohdr	2.1.4.2.3.12
get_from_racal	2.2.2.4.3
get_racal_time	2.2.2.4.7
get_start_date_fiohdr	2.1.4.2.3.15
get_start_time_fiohdr	2.1.4.2.3.19
get_tape_number_fiohdr	2.1.4.2.3.17
get_time	2.2.2.6.7
get_version_chksum_fiohdr	2.1.4.2.3.21
global.h	2.2.1.9
goto_handler	2.2.2.5.9
handle_remote_rtc	2.2.2.5.13
handle_rtc	2.2.2.5.7
handle_stdin_rtc	2.2.2.5.8

help_function 2.2.2.7.12
help_handler 2.2.2.5.10
info_field_bytes 2.1.4.2.3.7
info_handler 2.2.2.5.11
init_apdu 2.2.2.5.18
init_eotv_action 2.1.4.2.14.1
init_file_info 2.2.2.8.1
init_logger_for_copying 2.2.2.2.7
init_logger_for_playback 2.2.2.2.8
init_logger_for_recording 2.2.2.2.9
init_logger_for_server 2.2.2.2.6
init_logger_playback_state 2.2.2.2.17
init_logger_variables 2.2.2.2.4
init_pdu 2.2.2.5.19
init_queues 2.2.2.3.4
init_rtc 2.2.2.5.6
init_state_variables 2.2.2.2.3
init_user 2.2.2.7.13
ini_disk 2.1.4.2.10.4
ini_file_slots 2.1.4.2.2.1
ini_fiolib 2.1.4.2.2.13
ini_fnet 2.1.4.2.12.1
ini_tape 2.1.4.2.14.15
ini_tbuffers 2.1.4.2.14.12
is_next_tape_volume 2.1.4.2.14.6
itoa 2.1.3.2.1
libassoc 2.1.1
Libfio files 2.1.4.2
Libfio Interface files 2.1.4.1
logfilt.c 2.2.2.9
logfilt.h 2.2.1.10
Logger Time Library: liblogtime 2.1.3
logger.c 2.2.2.2
logger.h 2.2.1.2
logger_accept_pdus 2.2.2.2.12
logger_clock_tick 2.2.2.2.21
logger_connect 2.2.2.2.23
logger_continue 2.2.2.2.24
logger_copying_state 2.2.2.2.16
logger_disconnect 2.2.2.2.25
logger_error 2.2.2.5.3
logger_fsm 2.2.2.2.15
logger_get_activity 2.2.2.2.33

logger_get_remote_rtc	2.2.2.2.13
logger_get_state	2.2.2.2.32
logger_init	2.2.2.2.5
logger_init_cmc	2.2.2.2.11
logger_message_location	2.2.2.5.1
logger_msg	2.2.2.5.2
logger_op_error	2.2.2.2.22
logger_playback_state	2.2.2.2.18
logger_put_remote_rtc	2.2.2.2.14
logger_record_state	2.2.2.2.19
logger_reset	2.2.2.2.20
logger_seek_relative	2.2.2.2.26
logger_speed	2.2.2.2.27
logger_start	2.2.2.2.29
logger_status	2.2.2.2.31
logger_stop	2.2.2.2.28
logger_suspend	2.2.2.2.30
logtime.c	2.1.3.2
logtime.h	2.1.3.1
main	2.2.2.2.1
monthtoi	2.1.3.2.2
Network Interface Access: libnetif	2.1.2
new_queue_elt	2.2.2.3.2
new_rtc	2.2.2.5.14
new_tbq	2.1.4.2.14.7
next_available_tbuffer	2.1.4.2.14.13
next_disk_volume	2.1.4.2.10.1
null_fiohdr	2.1.4.2.3.23
null_tbq	2.1.4.2.14.11
offset_datetime	2.1.3.2.7
on_time	2.2.2.6.10
open_fio_files	2.2.2.2.10
open_UNIX_disk_file	2.1.4.2.10.2
parse_start_file_info	2.2.2.5.24
print_advice	2.2.2.2.37
print_command_line_help	2.2.2.2.36
process_parms	2.2.2.2.2
promote_status_info	2.1.4.2.14.5
prompt_user	2.2.2.7.11
protocol_of	2.2.2.5.4
queue.c	2.2.2.3
queue.h	2.2.1.3
queue_count	2.2.2.3.8

queue_of	2.2.2.3.1
racal.c	2.2.2.4
racal.h	2.2.1.4
racal_close	2.2.2.4.2
racal_open	2.2.2.4.1
racal_seek_to_time	2.2.2.4.12
racal_start_playing	2.2.2.4.10
racal_start_recording	2.2.2.4.9
racal_stop	2.2.2.4.11
racal_to_cmc	2.2.2.4.5
read_ast_handler	2.1.4.2.14.23
read_controller	2.1.4.2.10.10
read_controller	2.1.4.2.14.22
read_dbuffer	2.1.4.2.10.9
read_fiofile_header	2.1.4.2.10.8
read_fiofile_header	2.1.4.2.14.19
read_stream	2.1.4.2.14.21
read_tbuffer	2.1.4.2.14.20
receive_rtc	2.2.2.5.16
reinit_queues	2.2.2.3.5
reset_racal_time	2.2.2.4.8
reset_time	2.2.2.6.5
rtc.c	2.2.2.5
rtc.h	2.2.1.5
rtc_size	2.2.2.5.5
scan_handler	2.2.2.5.12
seconds_to_racal_string	2.2.2.4.6
send_clock_tick	2.2.2.5.29
send_rtc	2.2.2.5.17
send_to_racal	2.2.2.4.4
set_command_fiohdr	2.1.4.2.3.13
set_fast_time_factor	2.2.2.6.3
set_fileid_fiohdr	2.1.4.2.3.11
set_frame_size_fiohdr	2.1.4.2.3.18
set_tape_number_fiohdr	2.1.4.2.3.16
set_time	2.2.2.6.6
set_version_chksum_fiohdr	2.1.4.2.3.20
status_request_handler	2.2.2.5.25
strings_to_datetime	2.1.3.2.4
string_to_seconds	2.1.3.2.3
tape_close	2.1.4.2.14.36
tape_concat	2.1.4.2.14.17
tape_getrawfd	2.1.4.2.14.35

tape_open	2.1.4.2.14.16
tape_read	2.1.4.2.14.18
tape_seek	2.1.4.2.14.30
tape_seek_time	2.1.4.2.14.31
tape_write	2.1.4.2.14.24
timer.c	2.2.2.6
timer.h	2.2.1.6
time_string	2.2.2.6.11
tvolume_available	2.1.4.2.14.32
tvolume_starttime	2.1.4.2.14.33
unfreeze_time	2.2.2.6.9
unix_tape_action	2.1.4.2.14.4
user.c	2.2.2.7
user.h	2.2.1.7
user_putstr	2.2.2.7.10
util.c	2.2.2.8
util.h	2.2.1.8
vlaunch	2.2.2.7.15
volume_endtime	2.1.4.2.10.18
volume_starttime	2.1.4.2.10.17
write_ast_handler	2.1.4.2.14.29
write_controller	2.1.4.2.10.14
write_controller	2.1.4.2.14.28
write_dbuffer	2.1.4.2.10.13
write_fiofile_header	2.1.4.2.10.12
write_fiofile_header	2.1.4.2.14.25
write_stream	2.1.4.2.14.27
write_tbuffer	2.1.4.2.14.26
yes_or_no	2.2.2.7.9